

PalmGrid, an artificial intelligence approach to automate cylinder task detection

by

David Sze-Ming Cheng

CEng MHKIE PMP

MEng, University of Birmingham, UK (2001)

MBA, University of Warwick, UK (1990)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Neuroscience)

The University of British Columbia

(Vancouver)

April 2019

©David Sze-Ming Cheng, 2019

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

PalmGrid, an artificial intelligence approach to automate cylinder task detection

submitted by Cheng, Sze-Ming David in partial fulfillment of the requirements for
the degree of Master of Science
in Neuroscience

Examining Committee:

Tim Murphy, Department of Neuroscience
Supervisor

Nicholas Swindale, Department of Neuroscience
Supervisory Committee Member

Lawrence Ward, Department of Neuroscience
Supervisory Committee Member

Tim O'Connor, Department of Neuroscience
External Examiner

Abstract

Stroke is a common cause of permanent disability accompanied by devastating impairments. Motor, sensory and cognitive deficits are common following stroke, yet treatment is limited. Along with histological measures, functional outcome in animal models has provided valuable insight to the biological basis and potential rehabilitation efforts of experimental stroke. Developing and using tests that identify behavioral deficits is essential to expanding the development of translational therapies. Forelimb Asymmetry Task experiments – often called Cylinder Tests – are used to study the impact of ischemic stroke and its subsequent rehabilitation to contralateral limb movements of studying rodents. Through assessments on qualitative and quantitative aspects of vertical exploration to the Cylinder Wall, extent of locomotor asymmetry is evaluated [25-27].

Traditionally wall rearing assessments are evaluated through manual, stop-watch based measurements that require laborious observations. Methods that automate the process were attempted such as the use of hardware-based sensor detections that passively probes of touches on the sensor grid. In its various implementations, the sensor-based methods fail to specify the limb that rears the wall nor depict the ways forelimbs are coordinated during the rearing.

Advent of artificial intelligence (AI) algorithms, notably Deep Neural Networks (DNN), helps to extract posture and coordinates of forelimbs [21]. PalmGrid is the first attempt to exploit AI posture extraction algorithms – based on 50 layers depth in ResNet Deep Neural Networks [29] together with posture extraction algorithm DeepLabCut [24] – to automate the assessment process with 70% detection accuracy using robust, open-source software. Further improvements in deep neural networks precisions, such as increasing its depth or incorporating advanced posture extraction algorithms, will further enhance detection precisions. In this way,

we will have viable alternatives to conduct cylinder test experiments without suffering from extra cost burdens and complex calibrations.

Lay Summary

We have implemented an Artificial Intelligence (AI) software-driven, automated cylinder test method using basic laboratory apparatus and a Raspberry Pi video recorder. Upon one-time training that made the AI Engine¹ to profess forelimbs recognition competence, subject videos can then be analyzed to localize of their forelimb locations. These forelimb locations are then analyzed if vertical exploration (wall-rearing) occurred. Assessments based on cylinder tests of subjects demonstrate detection accuracy around 70%.

Utilizing software method bypasses several limitations of conventional approach. Firstly, laborious tasks of manual monitoring are no longer required. Secondly, calibrations of sensory networks to aid detections are spared, thus lowering equipment investments. Most importantly, any specific features of interests can be flexibly studied.

¹ AI Engine is made of an off-the-shelf Intel-based computer server with Windows 8.1 Enterprise software preinstalled with appropriate artificial intelligence and image processing software. For details, please refer to Section 2.3.

Preface

This dissertation is an original intellectual product of the author, David Sze-Ming Cheng. The fieldwork reported in the thesis was covered by UBC Animal Protocol Number A18-0321 and A18-0036.

The PalmGrid method is an Artificial Intelligent software-driven process to automate Cylinder Task experiments that are used to assess rodent behavior in stroke or rehabilitation. It is built on cutting edge Artificial Intelligence Deep Neural Network Algorithms (DNN) together with Posture Extraction and signal processing methods. The specific DNN algorithm used is ResNet-50, which refers to 50-layers of convolution neural networks made publicly available by Google in Tensorflow 1.1 software. ResNet-50 is the core foundation of posture extraction system called DeepLabCut developed by Adaptive Motor Control Laboratory from Rowland Institute of Harvard University Department of Neuroscience, which extracts cartesian coordinates of interested postures.

The PalmGrid process further capitalizes on DeepLabCut. It consists of an experimental setup of basic laboratory apparatus to record mouse behavioral videos and a server that tracked specific postures/features highlighted in the recorded videos. These extracted cartesian coordinates are then analyzed to discern of wall-rearing activities. In this way, data gathered in Cylinder Task experiments can be automated with minimal resources and investments.

Professor Tim Murphy of the University of British Columbia supervised the project, who initiated the requirements with Dr. Matilde Balbi to write the published manuscripts. I am responsible for design, implementation, experimentation setup, and testing using basic laboratory apparatus and single Raspberry Pi camera. I was much indebted to the help of Luis Bolanos and

Dr. Jamie Boyd for their assistance in building the transparent stools, and likewise for Dr. Balbi's contributions in practical implications of cylinder tasks experimentations. Last, not least, the help of Dr. Jamie Boyd who helped fast-tracking Raspberry Pi implementation required for the experimentation setup.

Table of Contents

Abstract	iii
Lay Summary	v
Preface.....	vi
Table of Contents	viii
List of Tables.....	xiii
List of Figures	xiv
Glossary.....	xvi
Acknowledgements	xvii
Chapter 1 Introduction	1
1.1 Cylinder Test.....	1
1.2 Different Variants of Cylinder Tests.....	3
1.3 Organization of this thesis	6
1.4 Wall-Rearing Detection using Touch-based sensing techniques	7
1.5 Brief History of Deep Neural Network and its measurement metrics	9
1.6 ResNet Algorithms.....	12
1.7 Deep Neural Networks Posture Extraction Algorithms	13
1.8 Research Aims	15
1.8 Our Design Prototype: PalmGrid.....	18
Chapter 2 PalmGrid.....	20

2.1	Introduction.....	20
2.2	Experimentation Setting.....	22
2.2.1	The Recording Apparatus.....	22
2.2.2	The PalmGrid Station.....	24
2.2.3	Choosing appropriate Artificial Intelligence Algorithmic configuration	25
2.3	Implementation	27
2.4	Detailed Methods	29
2.4.1	Training PalmGrid recognition capabilities.....	29
2.4.2	Localizing Forelimbs of Test Subjects	32
2.5	Errors introduced by AI and its relevance.....	34
2.6	PalmGrid Signal Processing Module	37
2.6.1	Harmonics filtering.....	37
2.6.2	Conversion to Polar Coordinates.....	39
2.6.3	Extract Slow-Moving Frames.....	40
2.6.4	Extract Coherent Fragments	40
2.6.5	Identify Congruence Points	41
2.6.6	Remove Outliers of Refined SubFragments	41
2.6.8	Gauge Refined SubFragments into wall rearing episodes	44
2.6.9	Export of wall-rearing episode	44
2.6.10	Compile wall-rearing episodes into video fragments	45
2.7	Experimental Testing	47
2.8	Results.....	48
2.8.1	Overall Results	49
2.8.2	Independent Assessment of Left and Right Forelimbs	52

2.9 Discussion	53
2.9.1 Correctly Recognized Touches	53
2.9.2 False Alarms (Positives) and Error Propagation Modeling	53
2.9.3 False Negatives or Missing detections	57
2.9.4 Different correct recognition and omission rates for left and right paws	58
2.9.5 Making use of Wall-Rearing Episode Report to enhance efficiency	59
2.9.6 Comparison of labor time required to use PalmGrid	60
2.9.6 Benefits of the software approach	61
Chapter 3 Design Choices and Discussions	62
3.1 Strength of PalmGrid	62
3.2 Limitations of PalmGrid	64
3.3 Future Improvement Areas	65
3.3.1 Use of more advanced artificial intelligence algorithms	65
3.3.2 Use of more advanced posture extraction algorithms	65
3.3.2 Dual-camera, epipolar geometry approach	66
3.3.3 Changing PalmGrid signal processing approach to machine learning	67
3.4 Concluding Remarks	68
Bibliography	71
Appendices	73
Appendix A – PalmGrid Hardware and Settings	73
A.1 Hardware Components	73
A.2 PalmGrid Experimentation Setup	75
A.2 Raspberry Pi Video Camera version 2 Specification	76
A.3 Raspberry Pi Video Recording Scripts	76

Appendix B – PalmGrid Signal Processing and Gauging Module Pseudocodes.....	77
B.1 Coherent Fragment Extraction.....	77
B.2 Congruence Points Identification.....	77
B.3 Refined Sub-Fragment Gauging	78
Appendix C – Test Results.....	79
C.1 Overall Left and Right Forelimbs taken together	79
C.2 Overall Left and Right Forelimbs taken together in Percentage.....	80
C.3 Right Forelimb Only in Number of Touch	81
C.4 Right Forelimb Only in % of Touch.....	82
C.5 Left Forelimb Only in Number of Touch	83
C.6 Left Forelimb Only in % of Touch.....	84
Appendix D: Methods for Forelimb Tests	85
D.1 Cylinder Test (Li & McCullough (2004)).....	85
D.2 Forelimb preference and sliding test (Shanina & Redecca 2006)	87
D.3 Cylinder Test (Schallert (2000))	89
D.4 Paw-Dragging Method.....	91
Appendix E: PalmGrid Station Installation Manual.....	92
E.1 Python 3.6.....	92
E.2 DeepLabCut.....	94
E.3 ImageJ.....	97
E.4 ffmpeg.....	98
Appendix F: Labeling Forelimbs of Mouse Images that trains DeepLabCut.....	99
Appendix G: Ground Truth Reports.....	101

G.1 Report compilation procedures	101
Appendix H: Matlab scripts	156

List of Tables

Table 1: PalmGrid Setup and Training Process	31
Table 2: PalmGrid Analysis and Signal Processing Process.....	33
Table 3: Sources of Error for ResNet-50, hence DeepLabCut.....	36
Table 4: Computed Wall-Rearing Results of Left and Right forelimbs.....	58

List of Figures

Figure 1: Cylinder Test Configuration.....	2
Figure 2: Typical Cylinder Test Assessment.	2
Figure 3: Summary of various Cylinder Test methods	4
Figure 4: Touch Sensing Detection System	8
Figure 5: Convolution Neural Network Schematics	11
Figure 6: Top-1 Error Error Rates of different Deep Neural Network Algorithms	12
Figure 7: The ResNet Algorithm.....	13
Figure 8: Top 5 Recognition Accuracy of ImageNet Challenge Winners	14
Figure 9 Basic Laboratory Settings of PalmGrid	19
Figure 10: PalmGrid Process Overview	21
Figure 11: PalmGrid's Recording Apparatus.....	22
Figure 12 Cylinder Test configuration to maximize light contrasts.....	23
Figure 13: Comparison of DNN Algorithmic Performance	26
Figure 14: PalmGrid Posture Extraction & Wall-rearing Detection Process	27
Figure 15: Example Wall-Rearing Hits Report.....	28
Figure 16: DeepLabCut Sample Training Photos that show the labeled forelimbs	30
Figure 17: An illustration of DeepLabCut misconstruing virtual image.....	35
Figure 18: PalmGrid Signal Processing & Gauging Filters	37
Figure 19: Conversion of DeepLabCut forelimb coordinates to PalmGrid language	39
Figure 20: Some postures that fulfill two coherences and congruence criteria.....	42
Figure 21: Removal of detected wall-rearing episodes	43
Figure 22: Sample Refined SubFragment Report	45
Figure 23: Synchronies of forelimbs in wall-rearing	46
Figure 24: Sampled Labeled Images	48

Figure 25: Test Results of PalmGrid for a cohort of n=5	51
Figure 26: PalmGrid Error Propagation Model.....	54
Figure 27: Lack of depth cue made resolution of wall-rearing challenging.....	56
Figure 28: Illustrative comparison of PalmGrid assessment time	61
Figure 29: Hypothetical PalmGrid design based on full-scale machine learning	67
Figure 30: Cylinder Test Score Calculation	86
Figure 31: Forelimb Activity Score Formula.....	87

Glossary

Major Terms of References	Description
Cylinder Test	Also called the spontaneous forelimb use asymmetry test, is frequently used to assess post-stroke limb use asymmetries in mice
DNN	Deep Neural Networks, a specific construct of Convolutional Neural Networks that become the prevalent industry standard for machine-based recognition of images and videos
DeepLabCut	An open-source, General Public Use licensed toolbox written by Adaptive Motor Control Laboratory of Rowland Institute, Harvard University; that built on Tensorflow and prevalent posture extraction algorithms to extract posture coordinates of identified features of interests in the video stream. Posture coordinates are given in cartesian coordinates relative to Field of View of the camera.
ILSVRC	ImageNet Large Scale Visual Recognition Challenges, an annual contest for competing Deep Neural Networks algorithms who were given a set of images and videos to discern into given (say 1,000) categories of objects. Competence of benchmark is assessed by Top-1 and Top-5 recognition accuracies.
Principal Component Analysis	Data Science techniques that extract maximum variances from a set of presented data
ResNet	Residual Neural Network, an open source software bundled in Google Tensorflow that is the winner of the 2015 ImageNet Challenge of computer object recognition. In 152 layers of the construct, implemented in September 2016 possesses a recognition rate under an optimal condition that achieves human “Top-5 recognition rate” capability of less than 5% error.
Tensorflow	An open-source, machine learning General Public Use toolbox provided by Google to facilitate low entry barriers adaption in deep neural networks algorithms

Acknowledgements

I would like to express my deepest gratitude to the teachings and mentorship of Professor Tim Murphy who led me to the field of Neuroscience and inspired me of an interesting experimental discipline that I have not known throughout my professional life as an engineer. By far, our Central Nervous System remains the least understood metaphor; and at times it remains mysterious. Because of our lack of understanding, options available to rescue brain injuries and neurodegenerations are limited, and much remains to be explored.

What intrigued my interests most is how advanced engineering techniques can be applied to aid neuroscientific explorations. In this thesis, optics and artificial intelligence are employed to track posture of the mouse in studying animal behaviors under ischemic stroke, to overcome resources and skill barriers of neuroscience laboratories who are less conversant with prevalent artificial intelligence methods. I used to perceive the experimental medicine discipline as another sphere of knowledge from mathematics and applied physics. Admittedly, this is an eye-opening experience.

Not least, I would like to thank Dr. Matilda Balbi who provided expertized contributions in cylinder tests; Dr. Jamie Boyd who contributed his phenomenal expertise in Raspberry Pi; and Mr. Luis Bolanos who assisted the experimentation setup throughout the project.

"Trust in the Lord with all thine heart, and lean not unto thine own understanding. In all thy ways acknowledge him, and he shall direct thy paths."

Proverbs 3:5~6

Chapter 1 Introduction


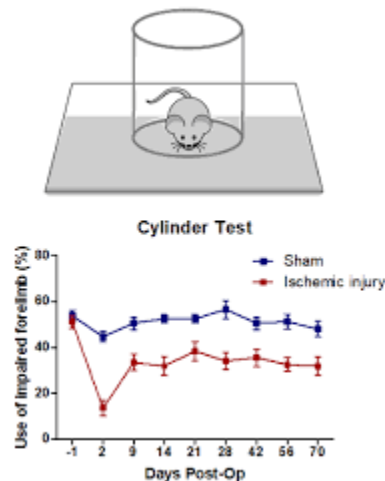
1.1 Cylinder Test

Stroke is a common cause of permanent disability associated with sensory and motor deficits. Developing and using tests for experimental stroke that helps to identify behavioral deficits is essential for the development of therapeutic interventions. The cylinder test, also called the spontaneous forelimb use asymmetry test, is frequently used to assess post-stroke limb use asymmetries in mice [25-27].

In a typical cylinder test, the mouse is put in an acrylic cylinder. The cylinder is mounted on a transparent stool where video equipment(s) is mounted underneath to record its activities (Figure 1). Current protocol mandates each recording to around ten minutes that can be separated by short time intervals for equipment calibration purposes to count respective left-forelimb, right-forelimb, or both-forelimbs wall rearing activities [25~27, also Appendix D.1]. Manual inspection is then required to identify the wall rearing episodes of respective forelimb on the cylinder wall, to assess the behavioral impact (Figure 2 and Appendix D.1). These wall rearing episodes refer to the time fragments where forelimbs rear the wall vertically. They are then expressed in different test scores and plotted to compare trends of locomotor asymmetry (lower plot, Figure 2).

Despite the test being relatively easy to perform, manual frame by frame counting in the number of touches within recorded video frames is laborious. Various methods had been attempted to automate the laborious and subjective collection of wall-rearing statistics in Cylinder Test. In a typical 4 minutes video recorded in 25 frames per second, the frame-by-frame review requires browsing of $4 \text{ (min)} \times 60 \text{ (seconds)} \times 25 \text{ (frames)} = 6,000 \text{ frames}$. Various

methods to conduct cylinder test assessments have been suggested to relieve the laborious exercise by employing slow play functions of image processing software such as VLC Player. An example of such is paw-dragging test [30], where at least three iterations to play the recorded mice videos in slow motions are required: first by skimming through the video of fragments in vertical exploration, followed by verifications, and subsequently to review particular wall-rearing fragment frame by frame to assess its quality of rearing. The need for second raters, prolonged monitoring (say 24 hours), and batch processing of different subjects' recordings further compound the time burden in analyzing Cylinder Test results.

	 <table><caption>Cylinder Test Data (Estimated from Graph)</caption><thead><tr><th>Days Post-Op</th><th>Sham (%)</th><th>Ischemic injury (%)</th></tr></thead><tbody><tr><td>-1</td><td>55</td><td>55</td></tr><tr><td>2</td><td>45</td><td>15</td></tr><tr><td>9</td><td>50</td><td>35</td></tr><tr><td>14</td><td>52</td><td>38</td></tr><tr><td>21</td><td>53</td><td>40</td></tr><tr><td>28</td><td>55</td><td>38</td></tr><tr><td>42</td><td>52</td><td>35</td></tr><tr><td>56</td><td>51</td><td>32</td></tr><tr><td>70</td><td>48</td><td>30</td></tr></tbody></table>	Days Post-Op	Sham (%)	Ischemic injury (%)	-1	55	55	2	45	15	9	50	35	14	52	38	21	53	40	28	55	38	42	52	35	56	51	32	70	48	30
Days Post-Op	Sham (%)	Ischemic injury (%)																													
-1	55	55																													
2	45	15																													
9	50	35																													
14	52	38																													
21	53	40																													
28	55	38																													
42	52	35																													
56	51	32																													
70	48	30																													
<p><i>Figure 1: Cylinder Test Configuration: Mice or Rats being put in laboratory cylinder to assess vertical exploration</i></p>	<p><i>Figure 2: Typical Cylinder Test Assessment. Image obtained from Li & McCullough [26] Chronic behavioral testing after focal ischemia in the mouse. Experimental Neurology Volume 187, Issue 1, May 2004, Pages 94-104</i></p>																														

1.2 Different Variants of Cylinder Tests

Different investigators employed varying score metrics to evaluate vertical exploration. As one of the first investigator proposing the method, Schallert [25] was interested to test the preference of rats in using non-impaired forelimb for weight shifting movements during spontaneous vertical exploration. This is why the test was devised to observe independent forelimb rearing as well as their respective landing during impaired and non-impaired conditions. Simultaneous forelimb rearing and landing were likewise recorded. There were no timing windows imposed for the test. The metrics are simple to understand and easy to be carried out.

Li & McCullough [26] slightly modified Schallert's method by focusing on forelimb use and rotational symmetry of mice. They are not interested in the landing counts but instead focus on forelimbs' vertical explorations. In this way, both independent and simultaneous rearing are recorded just as [25], plus they further define a metrics called "Right Forelimb Independent" that refers specifically to right forelimb staying on the wall during both impaired and non-impaired conditions. The mice are given 10-minute test period where 20 of these movement episodes are recorded. In the end, the independent forelimb movement and simultaneous counts during impaired and non-impaired conditions within the 10-minutes test period are given a metric score in the evaluation. Please refer to Appendix D.1 for detailed description of methods.

Shanina and Redecker [27] focused on recovery after photothrombotic infarcts in rats. In addition to the conventional independent and simultaneous forelimb rearing, they are also interested in wall sliding. The Forelimb Activity Index and sliding score percentage are computed, based on the formula given in Appendix D.2.

Recently, Roome and Vanderluit [30] noticed that the conventional methods of Cylinder Test were not sensitive enough for mice as compared to rats, judging by their lack of reliance on unaffected forelimb paw for postural support as compared to rats. Instead, they observed behavior termed "paw-dragging" where its impaired forelimbs drag along the cylinder wall rather than directly push off from the wall when dismounting from a rear to a four-legged stance. They are therefore interested to quantify the number of paw-drags and expressed as a percentage of total paw touches during an experimental session. Details of the method are depicted in Appendix D.4.

A summary of these four methods is given in Figure 3.

Features	Schallert et al 2000	Li & McCullough 2004	Shanina & Redecker 2006	Roome & Vanderluit 2015
Independent Left / Right Limb rearing	✓	✓	✓	✓
Independent Left / Right Limb landing	✓			✓
Simultaneous left/right limb for contacting / lateral movements	✓	✓	✓	✓
Simultaneous landing	✓			✓
Right Forelimb Independent		✓		
Sliding Movements			✓	
Lateral & Vertical Mvt			✓	
Independent Dismount				✓
Dragging				✓
Compute	Wall-assoc ratio Landing ratio	Cylinder Test Score	Forelimb Activity Score	Paw-Dragging Score

Figure 3: Summary of various Cylinder Test Metrics in various cylinder test methods as Schallert [25], Li & McCulloch [26], Shanina [27], and Roome & Vanderluit [30]

It is noted that while score metrics and features-of-interests (such as right forelimb independence, dragging etc. – see Figure 3) differ between various methods, the core principles remain unchanged. In order to automate these methods with maximum flexibility, wall-rearing detection that depicts the time and synchrony of these independent forelimb episodes is needed to gauge of the left, the right, and both forelimbs rearing. If particular method requires further data mining into particular rearing episode, it would be nice to have raw episodic data available to aid further analysis.

We have integrated these requirements into our prototype model to build a markerless cylinder test automation system called PalmGrid. In sum, PalmGrid is the novel approach that combines interdisciplinary excellence of Behavioral Neuroscience, Artificial Intelligence, Signal Processing, and Engineering. Its behavioral neuroscience requirements have been well adopted to assess locomotor asymmetry of rodents, especially in behavioral recovery studies of stroke as a result of plasticity in Central Nervous System [25-27, 30]. It will equally enhance efficiencies in conducting left-right forelimbs trait preference (biodiversity) experiments as prolonged observations are required to capture mice preference to grab food [33-34]. By leveraging advances in cutting edge artificial intelligence to locate the forelimbs and signal processing to make sense of them, Cylinder Test experiments can be automated with improved efficiencies that is easy to setup, easy to adapt, and easy to use by resources limited laboratories.

1.3 Organization of this thesis

This thesis is organized as follows. Chapter 1 refers to brief purposes of cylinder test and the corresponding history of automatic touch sensing development. Section 1.5 – 1.7 refers to a brief history of Deep Neural Networks that provides basic terms of reference; and Section 1.8 refers to research aim of PalmGrid that explains what we intend to utilize in Artificial Intelligence to assist us in automation of touch sensing in Cylinder Test. Given the interdisciplinary nature of current research, it is believed that a lengthier introduction in Section 1.5 depicting brief history of Artificial Intelligence will help readers to understand basic terminology and terms of references in the neuroscience research herein described.

Chapter 2 describes the features and design of PalmGrid. Section 2.1 briefly describes its design philosophy, and what is required to employ the system. Section 2.2 elaborates on experimental settings to conduct PalmGrid-based Cylinder Test experiment. Section 2.3 provides high-level implementation block diagram for the PalmGrid process, with section 2.4 detailing the steps underlying the process described in 2.3. Section 2.5 provides brief highlights into the less known stories of Deep Neural Networks despite its hype, leading to the necessity of signal processing and decision gauging algorithms of PalmGrid in Section 2.6. Remaining sections of the chapter discusses its algorithmic and process performance.

Chapter 3 opines on PalmGrid's strengths and weaknesses. Section 3.1 discusses its strength; while Section 3.2 outlines circumstances where the processes may not be applicable. There are many areas that the algorithm can be enhanced, and these are outlined in Section 3.3, including discussions of why certain technical options have opted while others not taken into consideration.

1.4 Wall-Rearing Detection using Touch-based sensing techniques

Previous attempts to automate Cylinder Test include the use of hardware touch sensing techniques. Touch sensing technologies are used in many applications such as smartphones, tablets, laptops, information kiosks, etc. Touch screens are very intuitive and easy to use; they also save space because their screen and interface are spatially integrated. Many touch sensing technologies have been developed for commercial purposes. Examples include technologies based on infra-red sensing elements [1–4], resistive [5,6] and capacitive sensors [7–9], cameras [10], the acoustic-based sensors [11–13], and others [14–16].

The mutual capacitive method is a popular touch-sensing approach that is extensively adopted in smartphones and tablets [17]. In this method, the touch interface is constructed of rows and columns of transparent tracks made of conductive paint. The row and column tracks are separated by a thin glass layer. Each row/column is electronically charged by an individual driver circuit. When the mouse touches the cylinder at a specific position where conduction tracks were laid, the capacitance at the intersection between the row and the column at this position changes; the point of pressure on the panel can thus be localized by scanning all the other non-energized rows and columns and computing the capacitance at all intersections, and recorded by a miniature Raspberry-Pi in further processing [18]. Most of the aforementioned touch sensors can be classified as active sensing techniques because touch detection depends on transmitting and receiving a signal that is perturbed by a touch.

In 2017, Kinsmen laboratories of University of British Columbia has setup the capacitive sensor infrastructure in an attempt to automate touch sensing. Capacitive paints were laid in a

tapered beam geometry that forms a grid network to detect rodents touches as its limbs touch the grid. These touches are then stored and analyzed using custom software written in Python. The settings achieved high sensitivity of 96.2% in some form of touch detection, *be it of bodies, limbs, or head*. Its advantage is high sensitivity in locating the touch.

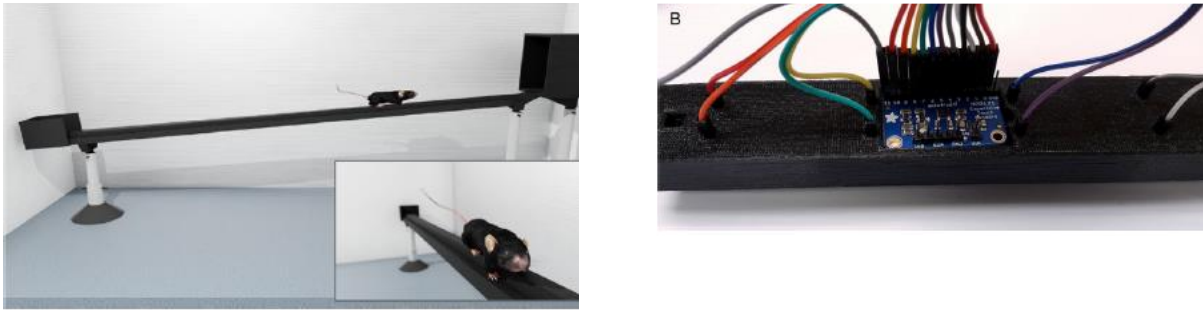


Figure 4: Touch Sensing Detection System, left panel displays the tapered beam geometry where capacitive paints were laid, while the right shows its DC electric connection to capacitance sensor chips. Images obtained from Fig 1 and Fig 2B in [19] Ardesch & Murphy. *Journal of Neuroscience Methods* 291 (2017) 221–226

Despite its hardwired accuracies, the method suffers from two major drawbacks. Other than the usual false positives due to latent touches of nearby sensors, it was impossible to discern whether the forelimbs or hindlimbs triggered the touch. Such deficiency poses challenges in the study of contralateral ischemic stroke that requires comparative study of left versus right forelimb mobilities impacted by stroke and corresponding rehabilitation studies. The method cannot fulfill wall rearing episodic extractions as required in various cylinder test protocols. It is also challenging to discern special features-of-interests such as “paw dragging” and “paw sliding” without explicit observations of forelimb synchronies in their rearing. Last but not least, the experimentation setup also requires costly investment and calibrations into touch sensing apparatus that are only affordable by laboratories with extensive engineering expertise in-house.

1.5 Brief History of Deep Neural Network and its measurement metrics

It is a broad consensus that one of the founding father of theoretical artificial intelligence was Alan Turing of Queen's College, Cambridge². With his works the Allied Forces had sped up to "brute force" the encryption code and deciphered the communications of German Navy to pave the victory of WWII. These algorithms soon evolved to become the cornerstone of modern machine learning methods. In his lecture in London Mathematical Society 1947, he opined that

"Let us suppose we have set up a machine with certain initial instruction tables, so constructed that these tables might on occasion, if good reason arose, modify those tables. One can imagine that after the machine had been operating for some time, the instructions would have altered out of all recognition, but nevertheless still be such that one would have to admit that the machine was still doing very worthwhile calculations. Possibly it might still be getting results of the type desired when the machine was first set up but in a much more efficient manner. In such a case one would have to admit that the progress of the machine had not been foreseen when its original instructions were put in. It would be like a pupil who had learned much from his master but had added much more by his own work. When this happens I feel that one is obliged to regard the machine as showing intelligence."

Turing, 1947

Since then machine learning has powered ahead with hallmark innovations in perceptron algorithms (Minsky & Papert et al 1969), Prolog programming language (Colmerauer et al 1972) and a host of others that spearheaded much of the artificial intelligence developments in the 20th Century. Their euphoria gave rise to a time of unrealistic expectations that subsequently dissipated as promises trailed behind hypes (The Economist 1992 September op ed quoting "Artificial Stupidity"). Thereafter development in the commercial arena, notably research funding, substantially shrank in the 1990s that left caring of this lonely child technology to be confined within universities' computer science laboratories.

In 2006, Professor Geoffrey Hinton of University of Toronto invented Deep Belief Net that became the first neural networks to learn decoded information state³ based on his understanding

² https://en.wikipedia.org/wiki/Alan_Turing

³ In computer science terminology, decoded information state is termed internal representations

of synaptic neurons architecture. Its breakthrough in object recognition rate – though modest in present day standards – revived commercial interests in machine learning that reopened enterprises’ R&D interests (such as Google and Microsoft) to substantially invest their research efforts into artificial intelligence sector. Microsoft Research and Amazon soon actively followed around 2010.

In brief, Deep (Convolution) Neural Network is a layered autocorrelator⁴ where each layer is responsible to discern specific feature sets of the data presented at its input, followed by an optional ReLU layer⁵ that makes decisions whether specific criteria are met. The stacking of these layers of autocorrelators effects adaptation of features-of-interests to specific patterns, such that any future presented data in similar pattern can be recognized [20]. For example, if an AI-Engine was trained to recognized of rodents forelimbs, then after iterations of training, it professes the knowledge to recognized such forelimbs of similar sizes and shapes.

⁴ Autocorrelator is an algorithmic process that correlates an input signal with a delayed copy of itself as a function of delay. The analysis of autocorrelation is a mathematical tool for finding repeating patterns that detects if the incoming signal resembles to features it looks for. An example of its application is facial recognition, where the shapes of particular person’s face is compared to a database of facial edges.

⁵ In the context of artificial neural networks, ReLU (or rectifier) is an activation function defined as the positive part of its argument:

$$f(x) = x + \max(0, x)$$

where x is the input to a neuron. This is also known as a ramp function and is analogous to half-wave rectification in electrical engineering.

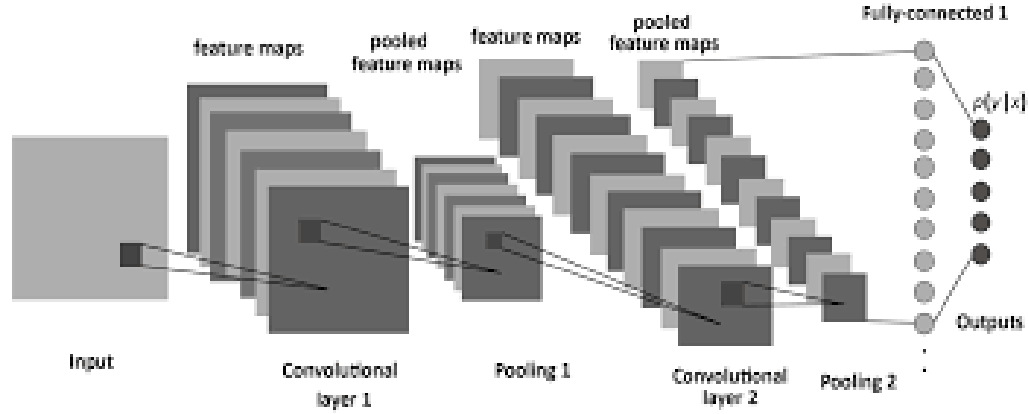


Figure 5: Convolution Neural Network Schematics. Image obtained from https://res.mdpi.com/entropy/entropy-19-00242/article_deploy/html/images/entropy-19-00242-g001.png

To benchmark the effectiveness of different DNNs, the ImageNet challenge was instituted as an industry effort to certify its performance since 2010. Soon the exercise became the platform of intellectual competition among big companies and research institutions alike such as NEC, MIT, Stanford, Microsoft, and Google. ImageNet presents portfolios of test images in different categories of stationary and moving objects to test out the accuracies of machine learning to recognize these objects.

The metrics to appraise effectiveness of Deep Learning algorithm is based on its recognition and localization capabilities. In terms of recognition capability, “Top-5 successful recognition rate” measures how accurate particular neural network is capable of discerning a particular image into respective categories irrespective of where it is in the presented picture⁶. “Top-1 recognition rate” was used to measure corresponding accuracies of the algorithms to “spot on” specific objects into specific categories, for example, “Chihuahua” in the category of “Dogs”. If the picture is recognized as “Wolf” or “Jackal”, their “Top-1 accuracy” will not score while its

⁶ For details of ImageNet competition, refers to https://en.wikipedia.org/wiki/ImageNet#History_of_the_database

“Top-5 accuracy” score will get one mark, and so forth. Figure 6 and Figure 8 shows different “Top-1” and “Top-5” recognition rates in respective winners of ImageNet challenges over the years 2010 - 2016. ResNet in different layers (34, 50, 101, 152) Top-1 recognition rate is also displayed although its 50-layers configuration was the winner in 2015.

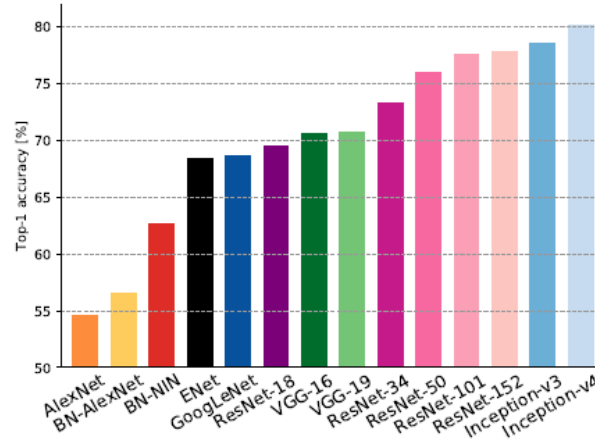


Figure 6: Top-1 Localization Error Rates of different Deep Neural Network Algorithms. Image obtained from Canziani & Paszke [21]. An analysis of Deep Neural Networks for practical applications. Computer Vision and Pattern Recognition April 2017

1.6 ResNet Algorithms

ResNet Algorithm was one form of Deep Neural Network invented by four scholars during their time in Microsoft Research in late 2015 [29]. It was known by then that constructs evolved from Deep Belief Network variants (and other planar networks such as VGG-19) suffer from major drawbacks in vanishing gradients, a symptom where learning errors saturated as stacking increase. Such metaphor renders learning of successive layers of artificial neurons to saturate, setting limits to depth of Deep Neural Networks [29], and hinders corresponding Top-1 and Top-5 error from further reductions.

The ResNet algorithm advances from plain networks' learning barrier by restricting the learning optimization within a few layers of neural networks. Its design ideas closely resemble the way animal cortex are organized into brain regions, with each one specialized and optimized in its own right while chained together to do big tasks (Figure 7). In ResNet, building blocks are cascaded together resulting in deeper architecture – hence higher recognition rate [29].

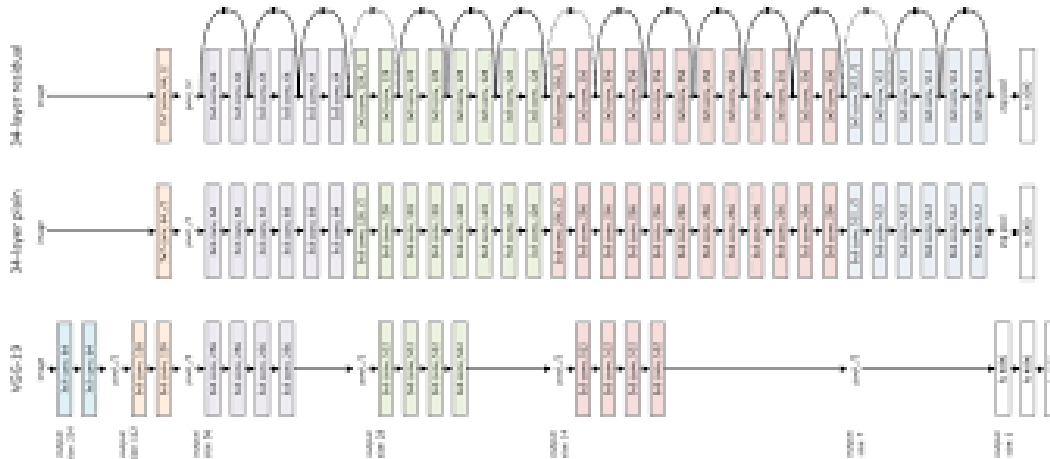


Figure 7: The ResNet Algorithm Architecture. Note that in contrast to other planar DNN algorithms like VGG-19, ResNet organized itself into smaller groups of artificial neuron layers for its self-contained optimization. This achieves faster and closer convergence. Image obtained from He & Sun [29]. Deep Residual Learning for Image Recognition. Computer Vision and Pattern Recognition. Dec. 2015,

1.7 Deep Neural Networks Posture Extraction Algorithms

Over the years since 2010, annual ImageNet Large Scale Visual Recognition Challenges (ILSVRC) demonstrated significant advances in image recognition capabilities of various DNNs; from 25% in “Top-5 recognition error”⁷ in 2010 to less than five percent in by 2015⁸ (Figure 8).

⁷ The Top-5 error rate is the percentage of test examples for which the correct class was not in the top 5 predicted classes. So, for example, if a test image is a picture of a Persian cat, and the top 5 predicted classes in order are [Pomeranian (0.4), mongoose (0.25), dingo (0.15), Persian cat (0.1), tabby cat (0.02)], then it is still treated as being ‘correct’ because the actual class is in the top 5 predicted classes for this test image.

ImageNet is an image base consisting of millions of images categorized into 1000 classes, top-5 error rate became benchmarks of efficacies of DNN algorithms.

⁸ For details on ImageNet Large Scale Visual Recognition Challenges, please refer to <http://www.image-net.org/challenges/LSVRC/>.

For the purpose of our discussion, ResNet-50 achieved 5.25% in recognition errors that approach human recognition capabilities. Further advances in ResNet to 152 layers by Microsoft Research in September 2016 claim less than 4.49% in Top-5 recognition error, exceeding human capabilities.

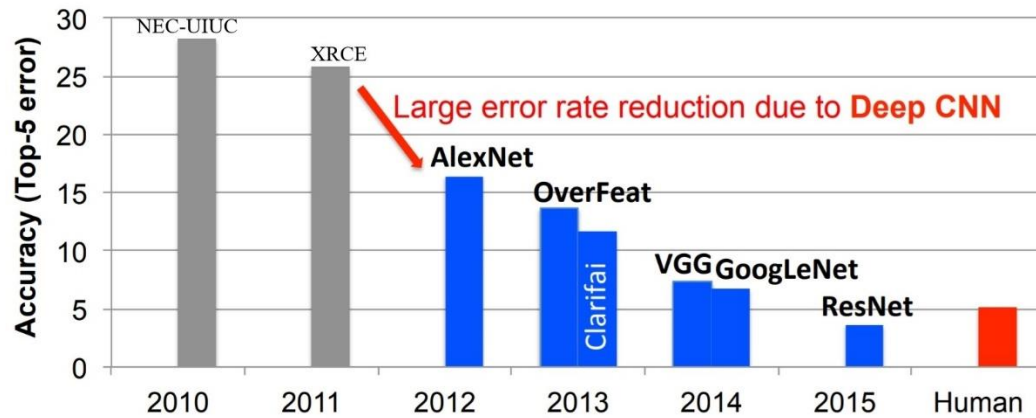


Figure 8: Top 5 Recognition Accuracy of ImageNet Challenge Winners. From 2012 onwards, Top-5 (Recognition) accuracies are approaching closer to human capabilities. ResNet was the first time human recognition capability was being challenged in 2015. Image obtained from Russakovsky & Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015

To facilitate widespread adoption of Deep Neural Networks, Google bundled the host of advanced DNN algorithms for general public use since late 2015. In 2017, Harvard University Adaptive Motor Control Laboratory leveraged ResNet-50 bundled in Google Tensorflow to integrate posture extraction algorithms into markerless pose estimation software toolbox called DeepLabCut [23]. Upon one-time training of DeepLabCut with a brief training video of what forelimbs look like, the software will extract forelimbs locations of experimentation videos⁹. These extracted locations are then analyzed to identify of wall-rearing episodes using custom-built signal processing algorithms.

⁹ For details of one time training of DeepLabCut, please refer to Section 2.4.1 Training PalmGrid recognition capabilities

1.8 Research Aims

We set out our research journey to investigate technical feasibility to automate Cylinder Test using frontier engineering methods, in particular, artificial intelligence and signal processing algorithms. We called this prototype project "PalmGrid" to symbolize its economics and simplicity: the two critical success factors for widespread adoption in laboratories with limited resources.

In particular, we are interested to answer the five research questions:

- 1) Can one automate Cylinder Test with only one simple Pi-camera under minimal calibrations?

The reason behind any automation was enhancing efficiency whilst lowering the costs of processing. PalmGrid objective was no different. We explore what can be done using the most basic Raspberry Pi Version 2 camera (Appendix A.2) and stretch its resolution to 1200 x 1200 pixels per frame¹⁰.

- 2) Can one achieve forelimb recognition in free-movement mice?

Given the fact that we are stretching resolution capabilities of DeepLabCut, forelimb recognition of free and fast-moving mice is also of interests. It would be easier to set camera focus to aid forelimb detection if we had head-fixed mice in similar experiments where bodily obstructions are not evident. In free-movement mice, maneuvering of subjects became a serious challenge to camera's shutter and focus capabilities that give rise to blurred images affecting

¹⁰ According to documentations of DeepLabCut [24], recommended maximum resolution of the software is 640 x 480 pixels resolution. Since forelimbs of mice was small, a finer resolution of 1200 x 1200 pixels would be required for adequate resolution of its features.

forelimb recognition. We are therefore interested to what extent can PalmGrid accepts free-movement mice to resolve and recognize forelimbs.

3) Can left and right forelimbs of subject mice be independently assessed?

Most importantly we are interested if left and right forelimbs can be independently assessed. From various methods depicted in Cylinder Test literature (Appendix D), independent assessment becomes the core principles to compute respective wall-rearing scores. Previous attempts to automate Cylinder Test process using mutual capacitive sensors failed to answer this question. Our fundamental assumption is that the ambient provides sufficient light that allows images to be resolved by the artificial intelligence algorithms to recognize forelimbs. Given the vague minimal lighting, is independent assessment feasible in artificial intelligence methods? In the context we are stretching our audacious goal: can we further design PalmGrid to capture activities such as wall-dragging or wall-sliding in prevalent protocols?

4) Can accurate locations of digits/forelimbs be tracked?

Even with independent assessment feasibilities, we need to be concerned with accuracies of these extracted coordinates as we need them, in various contexts, to compute metrics scores and gauges for wall-rearing episodes. Typical papers in artificial intelligence applications focus on “recognitions” to categorize different images, but in this context we need “spot on” precision accuracies to report where that features-of-interest is located. We want to understand to what extent of accuracies are analyzed out of these wall-rearing metrics, to translate these accuracies into the automation.

5) If 4) is feasible, can wall-rearing activities be discerned?

Last not least, and given all these constraints and coarse experimental settings, can we discern wall-rearing activities? We know that all images are coupled with noises that blur the accuracies of extracted postures. Thus if a machine is tasked to look into the extract postures as Cartesian coordinates, these distorted signals will misguide the algorithm from gauging actual wall-rearing activities. In our design of appropriate signal processing filters that extract these wall-rearing episodes, therefore, an estimate of its discernment accuracy is warranted.

The aim of this research is to capitalize on proven classification capabilities of cutting-edge artificial intelligence algorithms in extracting coordinates of the features of interests, to discern wall rearing touches in Cylinder Tests. We aim to test out the extent that prevalent machine learning and posture extraction approach can automate cylinder test touch sensing with modest laboratory setup.

1.8 Our Design Prototype: PalmGrid

With all the design criteria and constraints as set out above, we built our markerless Cylinder Test prototype called PalmGrid to test out various research questions raised in 1.7.

PalmGrid consists of a transparent base with a basic Raspberry Pi video camera mounted underneath. Free-movement mice are placed in a laboratory cylinder above the transparent base, so as to record mice activities above (Figure 9). Positioning the camera underneath minimizes the chance of forelimbs being obstructed from video recording due to body movements. The recorded videos are then analyzed by DeepLabCut AI Engine, followed by custom-built signal processing algorithms to identify wall-rearing. As left and right forelimbs are separately labeled, detection using PalmGrid approach allows left and right forelimbs to be independently tracked. Approaching automatic touch sensing using software approach by leveraging visible lights bypasses complexities in hardware setup, hence lowering the capital investments and skillset barriers that were otherwise prohibitive to resource-constrained laboratories.

The output is a wall-rearing report that details the timings in all wall-rearing episodes when respective left and right forelimbs reared the wall. Different investigators can then use this wall-rearing report to tailor their scoring as needed. If specific protocols require further data analysis into particular episodes to extract specific metrics, detailed episodic coordinates are also provided for their ongoing analysis. To facilitate investigators to review individual vertical explorations in its quality of rearing, each episode is compiled into smaller video fragments for their inspections.

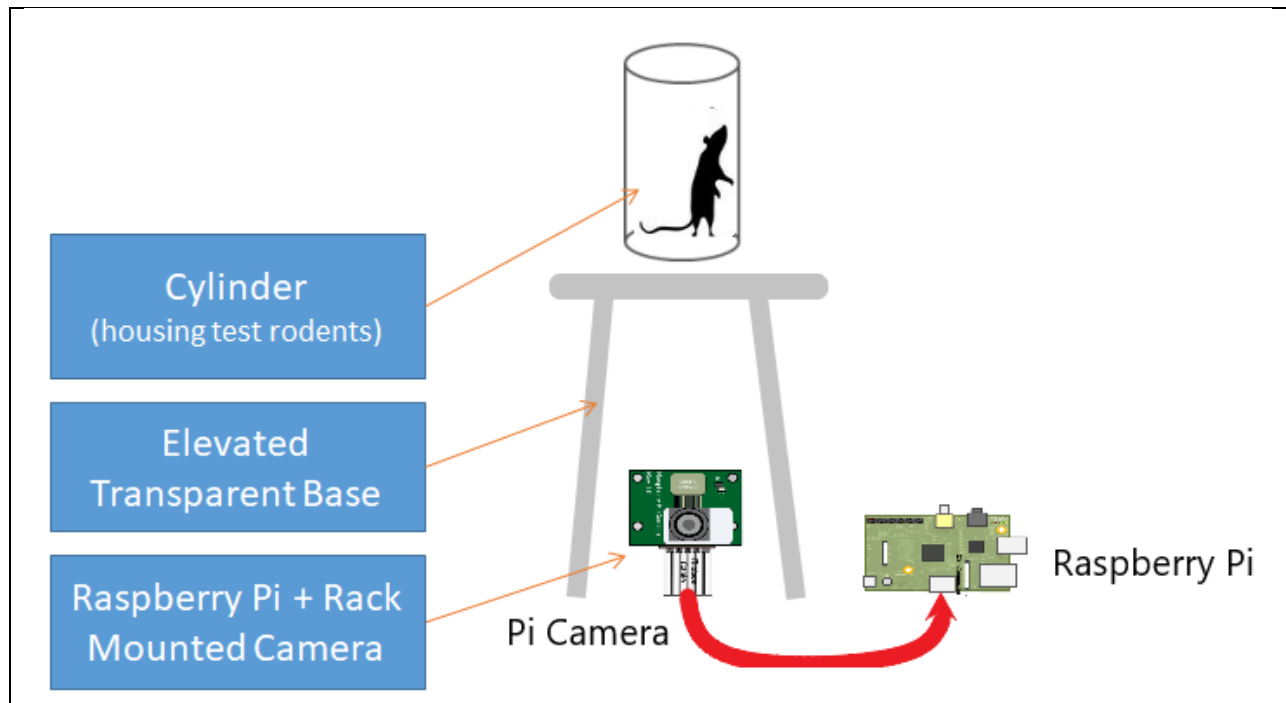


Figure 9 Basic Laboratory Settings of PalmGrid. The laboratory cylinder where mice are placed is mounted on a transparent stool. Underneath a Raspberry-Pi camera is placed to record videos of mouse activities. Recorded Videos are then analyzed by PalmGrid Station – a Windows 8.1 Enterprise server whose configuration is described in Section 2.2.2.

Chapter 2 PalmGrid

2.1 Introduction

PalmGrid is a process that capitalizes on DeepLabCut and Tensorflow's ResNet-50 to discern wall rearing episodes from predicted forelimbs locations¹¹. Video images of mice activities in the Cylinder Test are recorded in normal lights and analyzed using Tensorflow's ResNet-50 and posture extraction algorithms offered by DeepLabCut. In its hardware setting, the mouse is being housed in a laboratory cylinder supported by a transparent stool, underneath which a Raspberry Pi video camera is mounted to record mice activities bottom-up (Figure 11).

The system was deliberately designed for a single camera. Though multiple cameras system can utilize epipolar geometry methods to advance estimation of depth to higher precisions, we deliberately constrained our exercise to highlight the extent of achievements with prevalent DNN algorithms using simple Pi-camera with minimal ambience calibrations. For the same reason multiple camera systems that offer added dimensionality were not chosen, as adding further camera(s) will pose time synchronization problems between video clips besides increasing capital investments¹². In a similar rationale, sophisticated depth camera was not employed to gauge the depth of forelimb to calculate the location of wall rearing, in lowering of costs and calibration challenges as much as possible.

The PalmGrid System will be first trained to recognize what mice left and right forelimbs looked like and where they are located with respect to the referential coordinate system of video camera's Field of View in a one-time training process (see Section 2.4.1). Upon training in

¹¹ By episode it means a time fragment within which multiple cylinder wall touches had occurred.

¹² As at the date of writing, Raspberry Pi can only drive a single video camera in its basic settings. Having additional camera means adding another set of Raspberry Pi system that elevated the cost of equipment and challenges in synchronizing video times. As the experiment was set out to identify the lowest and simplest hardware setting for cylinder test, single camera approach has opted.

excess of 200,000 iterations¹³, the PalmGrid station professes posture extraction¹⁴ competence in video camera's referential coordinate system. One can then feed appraisal videos – of different studying subjects – into the PalmGrid station to predict forelimbs locations with respect to the cartesian coordinates in camera's Field of View. These predicted forelimbs locations are then processed by PalmGrid signal processing module to gauge wall rearing episodes (Figure 10).

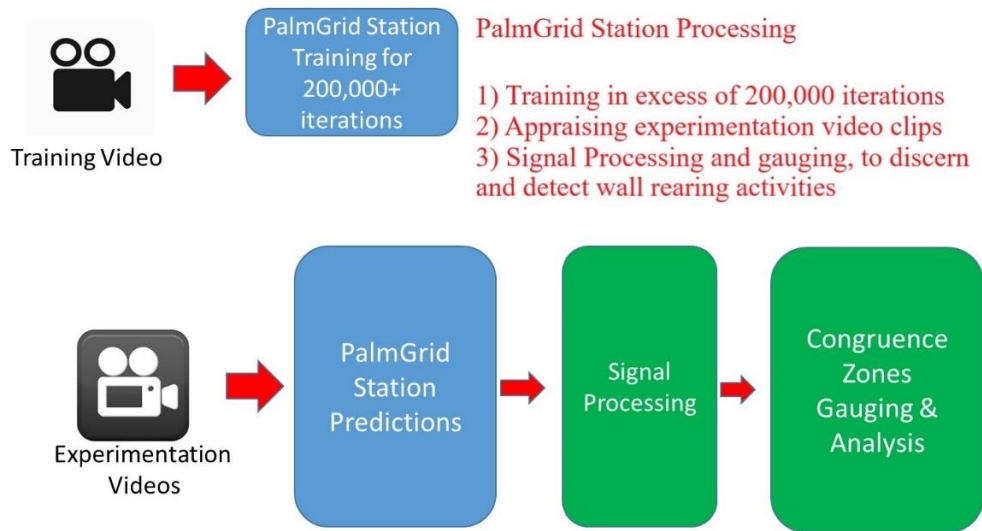


Figure 10: PalmGrid Process Overview, as detailed in Section 2.4

Observations are made to deduce criteria of forelimb locations as wall-rearing. It is noted that forelimb digits will first slowdown as its paw approaches the wall, followed by extending their distances furthest from the center of the cylinder in forward limb stretching until some of them are obstructed by the cylinder wall. The forelimb(s) will either find support on the cylinder

¹³ Training in excess of 200,000 iterations was a subjective call, to lower prediction errors to a plateau below 10^{-5} . Lower iterations will achieve higher order magnitude of error, further increasing prediction errors. In DeepLabCut, decent recognition competence is professed after a minimum of 64,000 iterations

¹⁴ Posture extraction is a buzzword in artificial intelligence discipline that refers to recognition and localization of specific features of interests such as limbs and the way subject stands in a given video. In our context, it refers to recognition and localization of digits in left and right forelimbs.

wall, and subsequently retrace from the wall. Because their forelimbs are obstructed, alteration in movement trajectory.

2.2 Experimentation Setting

Minimal hardware and calibrations remain core consideration in PalmGrid design philosophy, as it was devised with ease of adoption in mind. The hardware setting of PalmGrid System thus consists of a recording apparatus where wall rearing activities of mice are recorded and a PalmGrid station where recorded videos are analyzed. We did not assume laboratories are well resourced to profess lighting and raspberry Pi calibration expertise to engage in PalmGrid.

2.2.1 The Recording Apparatus

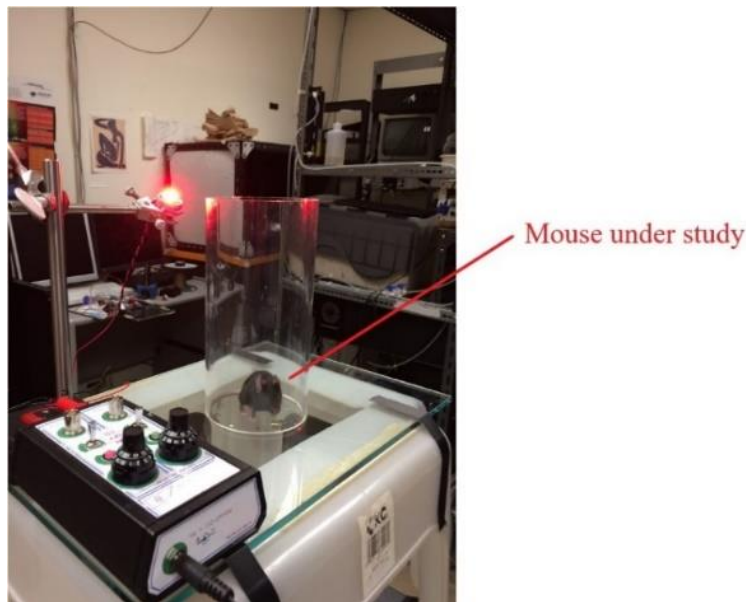


Figure 11: PalmGrid's Recording Apparatus

The PalmGrid's recording apparatus consists of basic laboratory equipment of a cylinder where the mouse is being housed and studied. The cylinder is being mounted on a transparent stool, underneath which a Raspberry Pi video camera is being mounted (Figure 9 & Figure 11).

To facilitate video recording of discernable quality, white cardboard with black top enclosure encase the cylinder, gathering as much room lighting as possible (Figure 12). Basic Raspberry Pi camera is configured to its maximum resolution mode, namely 1200 x 1200 pixels, to allow for its finest features (such as forelimbs) tracking possible with the basic camera. Lowering such resolution implies coarser pixel resolutions, impacting accuracies of feature identifications and hence the subsequent detection finesse.



Figure 12 Cylinder enclosed in white cardboard with black tops maximizing light contrasts

As mentioned, more sophisticated configurations such as depth camera and multiple camera systems were not opted, so as to assess what can be achieved with artificial intelligence under basic hardware configurations.

2.2.2 The PalmGrid Station

The PalmGrid station is a Windows 10 Server installed with at least 32GB of Memory and a prevalent Graphics Processing Unit from Nvidia that performs analysis of captured video images in Cylinder Test. These graphics processors determine how fast the intended features – such as left forelimb of the mouse – can be learned and analyzed by ResNet-50. The PalmGrid station is installed with the following application software, mostly General Public Use licensed software in artificial intelligence and image processing.

- Python 3.6 where DeepLabCut software runs on; and
- ImageJ that labels training video of the particular features to be recognized; and
- A video format converter that converts recorded video from de-facto Raspberry Pi's H264 to avi video formats to integrate with DeepLabCut; and
- Matlab where extracted forelimbs' coordinates are further analyzed into wall rearing episodes; and
- Microsoft Excel helps to tabulate final detection results for presentation¹⁵;
- Last but not least, a video to photo converter called “FFmpeg” that converts videos into photos and vice versa

Installation of these software followed respective configuration guides as recommended by respective vendors. Please refer to Appendix E for details.

¹⁵ Excel was chosen because of its easy to use without requirements to understand database administration and operations. Again, we intend to lower adoption barrier. More resourceful laboratories can consider other forms of organized storages, such as Oracle database management system.

2.2.3 Choosing appropriate Artificial Intelligence Algorithmic configuration

DeepLabCut offers two optional Artificial Intelligent algorithms that one can opt for: ResNet in 50 layers and 101 layers configurations. Choosing which ResNet configuration require our understanding how to balance recognition and localization accuracies of artificial intelligence algorithms.

Artificial Intelligence algorithm can be best understood with an analogy of visual cortex information streams. In our visual cortex, ventral (“What”) stream is responsible for recognition while the dorsal (“Where”) stream is accountable for localization. Likewise, artificial intelligent algorithmic performance is measured in terms of its recognition (i.e. “What”) and localization (i.e. “Where”) capabilities. Recognition capabilities metrics is depicted in “Top-5 accuracies”, while localization capabilities is appraised in “Top-1 accuracies”. Top-5 accuracy was widely reported to educate the public that artificial intelligence algorithm has already exceeded human recognition capabilities ever since the arrival of ResNet in 2015.

For PalmGrid that capitalizes on features of DeepLabCut toolbox that uses ResNet to provide recognition (Top-5) accuracies of over 95% as shown in Figure 8, further decisions are to be made in the choice of 50- or 101-layers configurations. Since Cylinder Test protocols require localization of forelimbs, this implies that we have to understand how accurate its localization capability is and balance such requirements against computational demand of Graphics Processing Unit in the captioned automation to achieve acceptable time to analyze each experimentation video. In doing so, realistic decisions can be made in what type of computing hardware is required to provide adequate localization accuracies within reasonable computational time.

The localization performance (Top-1 accuracy) chart as shown in Figure 13 is rarely reported beyond artificial intelligence discipline. For the two ResNet configurations, ResNet-50 offers 76% localization (Top-1) accuracy with 8 giga-computations in one forward computation parse; while in ResNet-101 a 78% localization accuracy shall require 25 giga-computations in one forward parse. This means that if we were to choose ResNet in 101 layers of depth configurations, we have to purchase a Graphics Processing Unit that possesses at least three times as much processing power as the corresponding ResNet in 50 layers depth for the same processing time in a given experimental video recording. In our laboratory for example, we have a Nvidia Quadro Graphics Processing Unit for the project, which was not very fast as compared to other models such as Nvidia Titan XP¹⁶. As a result, we have made a compromise to choose ResNet with 50 layers of depth that provides acceptable analytical time performance against a 76% localization accuracy¹⁷.

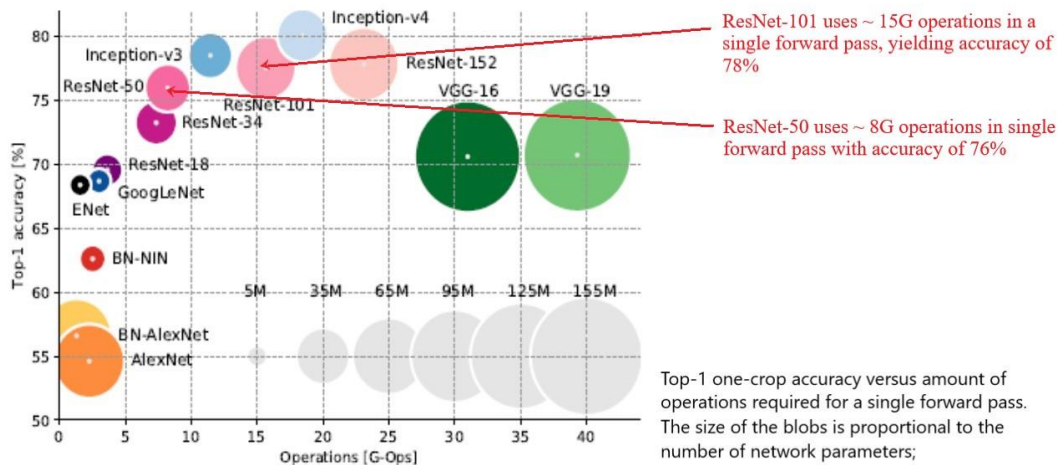


Figure 13: Comparison of DNN Algorithmic Performance. Image obtained from Canziani & Paszke [21] An analysis of Deep Neural Networks for practical applications. Computer Vision and Pattern Recognition April 2017

¹⁶ For specifications of Nvidia Quadro and Titan models, please refer to their specification sheets on Nvidia web sites: <https://www.nvidia.com/en-us/design-visualization/quadro-desktop-gpus/> and <https://www.nvidia.com/en-us/titan/titan-xp/> respectively

¹⁷ In other words, the algorithm has an inherent localization error of 24%.

2.3 Implementation

The recorded subjects' videos are first converted to PalmGrid acceptable video format to extract mice forelimbs' locations in cartesian coordinates. These forelimbs locations are then made sense by PalmGrid signal processing module that filters confounding noises introduced by Artificial Intelligence algorithm (see Section 2.5 and 2.6) to discern of wall-rearing hits. These wall-rearing hits are reported in the Wall-Rearing Episode Report in numeric tabular format, and as video reports.

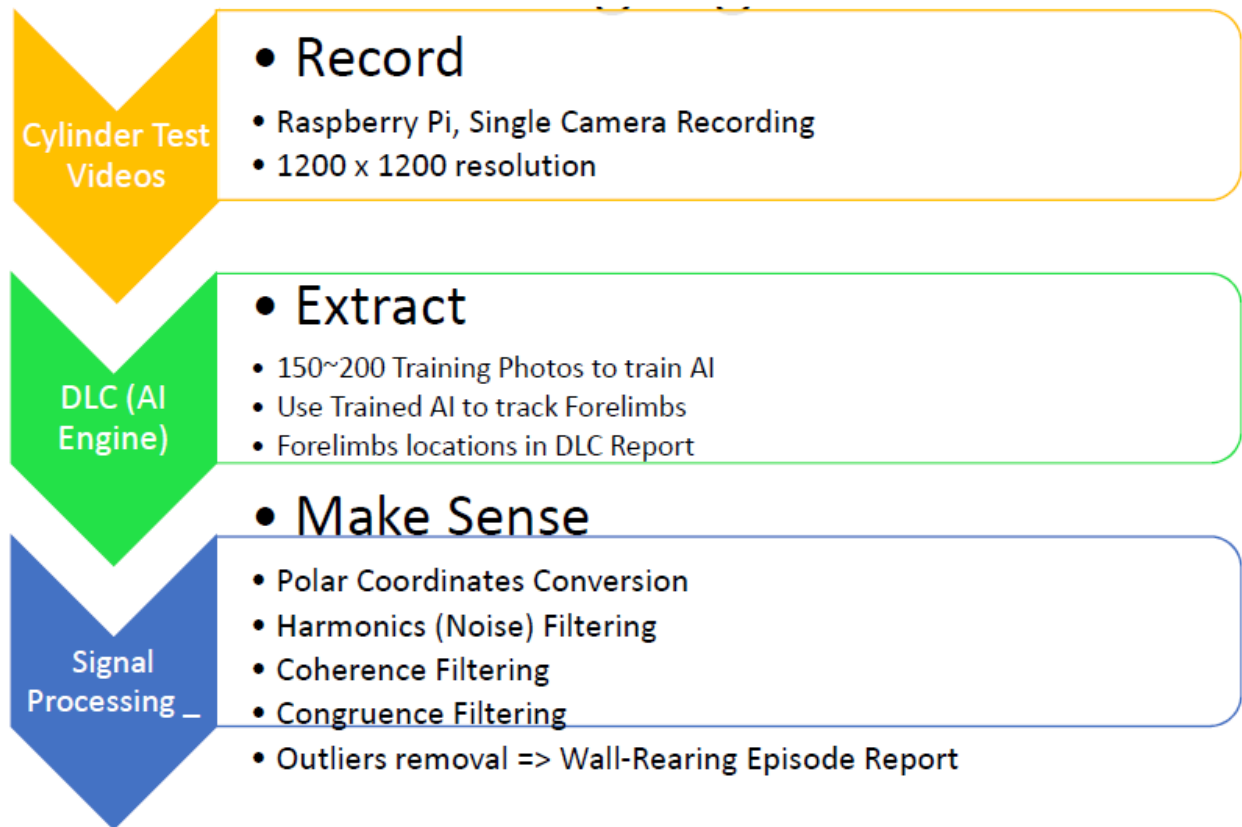


Figure 14: PalmGrid Posture Extraction & Wall-rearing Detection Process

After posture extraction, PalmGrid mines into the extracted posture database to identify wall-rearing episodes. Once we have got rid of the confounding noise (see Section 2.5 and

2.6.1) introduced by DeepLabCut, a series of signal processing filters are utilized to identify wall-rearing events.

The Wall-rearing episode report depicts respective time fragments within which wall-rearing most likely occurred. An example is shown in Figure 15. In here, the red underline refers to one of such episode in Fragment 2, where video frames 498 to 528 contains at least 1 wall-rearing touch(es). As the video was recorded in 25 frames per second, it means that the particular episode happened between 20 seconds (being $498 / 25$) and 21 seconds (being $528 / 25$).

Fragment Number	Start Fragment		End Fragment		Decision
	SubFragment	Midpoir	Start SubFragment	End_SubFragment	
1	8	362	-	-	-
-	236	137	335	1	-
Fragment 2	2	364	760	-	-
	-	513	498	528	1
	-	637	611	663	1
	3	762	1,180	-	-
-	803	792	814	1	-
-	1,039	937	1,140	1	-
4	1,187	1,203	-	-	-
5	1,210	1,218	-	-	-
6	1,221	1,349	-	-	-
-	1,287	1,239	1,334	1	-

Figure 15: Example Wall-Rearing Hits Report. Here Fragment 2 refers to a time period when a forelimb has one of its digits moving below 5 pixels per frame. Within this fragment, two plausible wall-rearing episodes are identified, the one underlined in blue refers to a wall-rearing episode between frame 498 to 528.

Investigators can then use these wall-rearing episodes to mine their cylinder test scores according to their intended protocols. To offer investigators with more flexibilities, the raw data of wall-rearing episodes are also stored in the Wall-rearing episodes report that facilitates them to perform specific data mining to derive intended metrics, such as “paw-dragging” in [30]. Videos of these discerned wall-rearing episodes are also compiled to help investigators assessing qualitative aspects of Cylinder Test requirements.

2.4 Detailed Methods

As mentioned in Figure 10, the PalmGrid station was trained of left and right forelimbs followed by analysis of subjects' videos. The end results of the analysis would be predicted forelimb locations that depicts coordinates of the forelimbs' digits and palms with respect to referential cartesian coordinates of video camera's Field of View. Forelimbs locations were then analyzed by PalmGrid signal processing filters to discern of wall rearing activities.

The detailed process of setup, training, analysis of forelimbs locations together with identifying the wall-rearing episodes are detailed below. Other than the training process that requires video recording to train the recognition capabilities of the artificial intelligence engine, the entire process was coded in Matlab to enhance code readership by neuroscientists.

User will first prepare a file that tells PalmGrid the respective locations of videos, extracted posture coordinates, and their intended locations of wall-rearing episode report. PalmGrid reads the batch file to generate the wall-rearing episode reports of all the videos in one go. For a batch of 10 videos of 4 minutes each, the serial batch processing took less than 20 minutes.

2.4.1 Training PalmGrid recognition capabilities

A training video was first recorded to train ResNet-50's recognition capabilities of forelimbs in the Cylinder Test settings. To facilitate learning of ResNet-50 in what mice left and right forelimb looks like, their locations in respective pictures in training video will be labeled using prevalent off-the-shelf image analysis tools – such as ImageJ¹⁸.

¹⁸ ImageJ is a public freeware of image processing, downloadable at <https://imagej.nih.gov/ij/>

To test whether PalmGrid could generalize forelimb recognition capability, we chose a different mice strain to train ResNet-50. In this arrangement a male AI-94¹⁹ mouse was placed in the cylinder to record its video activities. All animal procedures were approved by the University of British Columbia Animal Care Committee and conformed to the Canadian Council on Animal Care and Use guidelines.

Around 150~ 200 photos of mice forelimbs were labeled to train ResNet-50 that possessed varieties of postures. A few training photos are shown in Figure 16. DeepLabCut was supposed to accept a maximum of 800 x 600 pixels pictures [24]. But since forelimbs were so small as shown, we needed a higher pixel resolution that exposed finer features of forelimbs for recognition and localization. We had therefore stretched camera resolution to its maximum 1200 x 1200 resolution²⁰.

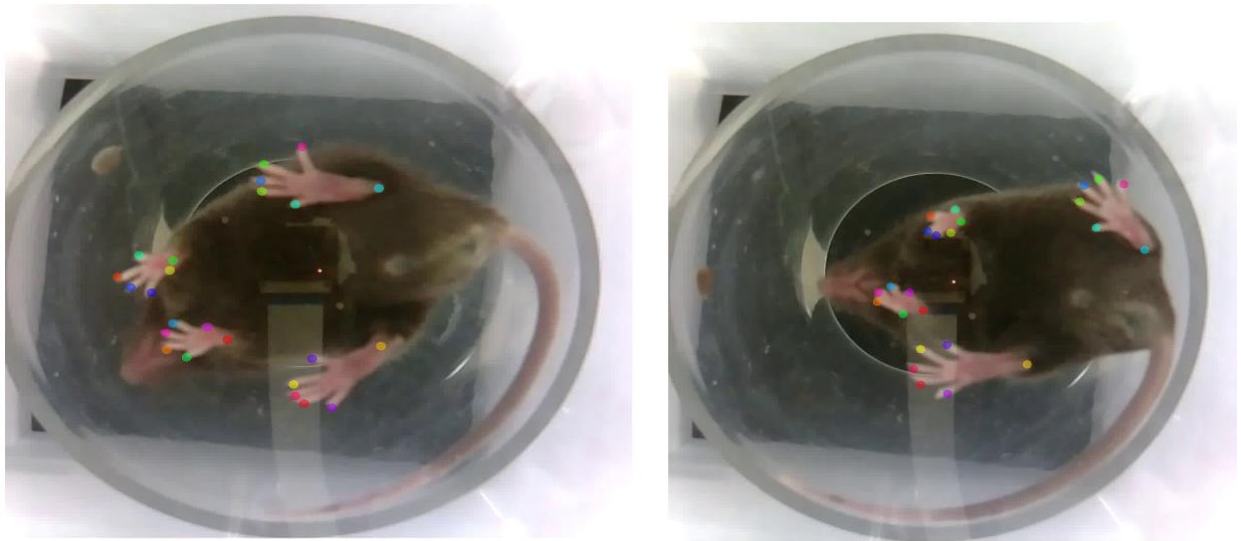


Figure 16: DeepLabCut Sample Training Photos that show the labeled forelimbs

¹⁹ Jackson Laboratory Stock No: 024115 | Ai94(TITL-GCaMP6s)-D;CaMK2a-tTA

²⁰ In a separate experiment, I have tried to conduct the training in 800 x 600 resolutions. The trained ResNet-50 exhibited poor recognition and localization capabilities, suggesting a higher resolution was needed.

Time taken to train ResNet-50 recognition capability was contingent on the configurations of machine learning platform and computation power of Graphical Processing Unit. Artificial intelligence engineers usually referenced best practices to determine the number of iterations required in specific applications. In here forelimbs postures accuracies were important, we followed recommendations from DeeperCut [31] in human posture recognition. DeepLabCut did not provide corresponding recommendations in similar applications [24].

Steps	Description
Phase I	Preparation and Training of PalmGrid Recognition Capabilities
1	Setup Cylinder Test configuration as in Figure 9. Focus Pi-camera to the upper-midline of the Cylinder for better image resolution in free movement activities
2	Record training video of a mouse to train AI-Engine of what forelimbs of mice look like. For my experiment, I have let the training mice to settle in the cylinder for 10 minutes before recording its activities in the cylinder for 4 minutes. Select a video frame of 1-minute duration with as many wall rearing activities as possible. Try to avoid video frames containing mouse grooming at this point.
3	Feed the training video into ImageJ System to generate around 200 training images for Labeling that exhibit clear digits and palm features.
4	For each training images, label both Left and Right Front Paws in all its fingertips and palm (i.e. six points per forelimb, total 12 points). As an illustration in forelimbs labeling, please refer to Appendix F. The process of forelimb labeling will depend on users proficiency in ImageJ. In my configuration, it took me 45 minutes to label the 200 images.
5	Train DeepLabCut with the labeled training images. A good iteration cycle of 200,000+ iterations is recommended ²¹ . Then evaluate the trained video as per instruction given in DeepLabCut user manual. This will typically take a few hours contingent on the GPU used. Upon completion of training, the PalmGrid Station professes left and right forelimbs recognition and localization capabilities. The system is now ready to appraise experimental videos.

Table 1: PalmGrid Setup and Training Process

²¹ Training 200,000 iterations is recommended by DeeperCut [31] – the predecessor of DeepLabCut [24]. The number of training iterations have always been a subjective call. DeeperCut calls for 82% localization accuracies in human posture extraction, and in their experience with ResNet-50 they recommend 200,000 iterations to allow AI-Engine to profess such localization accuracy.

2.4.2 Localizing Forelimbs of Test Subjects

All test subject videos were then prepared and saved in designated DeepLabCut folder for posture extraction [24]. These batches of video images were analyzed of forelimb locations in scripted command. Once the posture coordinates were extracted, they were then fed into PalmGrid signal processing module that identified wall-rearing episodes among presented locations.

A typical run to localize forelimbs in a 4-minutes video under Nvidia Quadro GPU was less than 20 minutes in our PalmGrid Station. For a typical batch of 10 videos, we left the PalmGrid script execution overnight in posture extraction. The process of test subjects' video recording, video image analysis, and corresponding signal processing to make sense of these coordinates are detailed below in Table 2. The entire process was pipelined in single command for large volume of video analysis.

Phase II	PalmGrid Processing
1	Record subjects' videos into PalmGrid station to localize forelimbs locations, with respect to video camera's Field of View.
2	Feed forelimb locations into signal processing modules of PalmGrid Station ²² .
3	Run PalmGrid signal processing module to analyze wall rearing episodes. The script, composed of 8 signal processing filters, run in one parse in the following steps
	<ul style="list-style-type: none"> a) Perform signal smoothing, to smooth out-of-range predictions of DeepLabCut²³; followed by b) Convert Extracted Posture Coordinates into polar coordinates based on the measured location of cylinder center²⁴; followed by c) Extract Slow Moving Fragments, based on Extracted Posture Coordinates that moves less than 5 pixels between successive frames²⁵; followed by d) Extract Coherent Fragments, based on tracking of minimal two slow moving digits that moves towards cylinder wall and subsequently retrace from it²⁶; followed by e) Identify Congruence Points, based on tracking of minimal distance between individual fingertips and palm of each forelimb²⁷; followed by f) Match Coherent Fragments with Congruence Points, into Refined SubFragments that refers to slow movements of changing trajectory²⁸; followed by g) Compute statistics for each Refined SubFragment and remove outliers that does not correspond to wall-rearing²⁹; and finally h) Consolidate this smaller subset of Refined SubFragments into episodes of wall rearing³⁰

Table 2: PalmGrid Analysis and Signal Processing Process

²² For each experimentation video, there are slight variations in the location of cylinder center with respect to the video camera's cartesian coordinates system. Simple photos tools such as Microsoft Photos can be used to depict the approximate location of the center of the cylinder. Approximate location of cylinder center is important to convert the cartesian coordinates into polar coordinates system aiding wall-rearing detections.

²³ Refer to Section 2.6.1 for details

²⁴ Refer to Section 2.6.2 for details

²⁵ Refer to Section 2.6.3 for details

²⁶ Refer to Section 2.6.4 for details

²⁷ Refer to Section 2.6.5 for details

²⁸ Refer to Section 2.6.6 for details

²⁹ Refer to Section 2.6.7 for details

³⁰ Refer to Section 2.6.8 for details

2.5 Errors introduced by AI and its relevance

While ResNet-50 is capable of matching human recognition capabilities expressed in Top-5 classification, PalmGrid also requires forelimb localization³¹ as benchmarked in [21]. In general, DNN algorithms that utilized ResNet-50 machine learning methods suffer from three major sources of non-linear errors. These errors are broadly termed as Max-Pooling³², ReLU³³, and Picasso³⁴ (Pooling) errors.

1. In Max-Pooling, the most likely cubic that resembles specific feature set pertaining to particular artificial neuron is chosen while others are ignored. Therefore in pictures where left thumb appears in several nearby locations (e.g. virtual image of forelimb due to acrylic reflection of cylinder), the sharper one is taken;
2. In ReLU, which is essentially a decision function for each feature set where non-linear error sources are introduced. In engineering, feeding successive images (i.e. signals) into the non-linear activation function results in uneven amplifications, resulting in harmonics noises³⁵;
3. In Picasso (Pooling) errors, much like the famous painter Picasso, the machine learning algorithm focuses on the specific feature set they look for without rationing its reasonableness. In our context, the thumb of left paw may be identified to attach to right forelimb; or an internal reflection of the cylinder can be mistaken as the actual thumb.

³¹ In DNN, precision refers to Top-1 recognition capabilities, instead of Top-5 recognition capabilities in ImageNet Challenges. For details, refer to [21].

³² For definition and illustrated examples of max-pooling, refers to https://computersciencewiki.org/index.php/Max-pooling/_Pooling

³³ ReLU is the abbreviation of rectified linear unit, which applies the non-saturating activation function. It effectively removes negative values from an activation map by setting them to zero. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. For details and illustrations, refer to https://en.wikipedia.org/wiki/Activation_function

³⁴ For definition and illustrated examples of pooling, refers to https://computersciencewiki.org/index.php/Max-pooling/_Pooling

³⁵ For discussions of activation functions and its relationship with harmonics noise, refer to <https://arxiv.org/abs/1603.00391>

The noise sources introduced in ResNet-50 will then predict forelimb digits and palms locations in the wrong place. For example, if the acrylic reflection of right palm is sharper than the actual one, DeepLabCut will misconstrue the virtual image as shown in Figure 17. Therefore signal processing filters are needed to correct these errors as far as possible. Likewise, ReLU errors are also observed with jittery digits between frames despite the forelimbs are resting on cylinder floor.

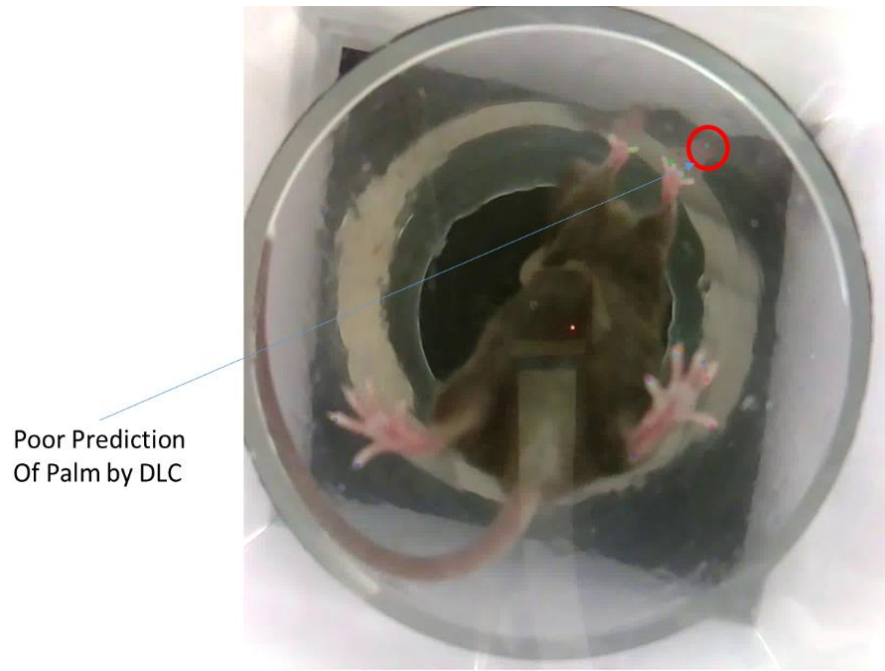


Figure 17: An illustration of DeepLabCut misconstruing virtual image of right forelimb palm as the actual one

Details of sources of errors introduced by ResNet-50 is tabulated in Table 3.

Sources of Errors	Causes of such error	Impact on Digits and Palms localizations
Max-Pooling	<p>Deep Neural Networks segregate images into smaller rubrics and auto-correlate each rubric to discern the closest resemblance to learned postures. This is very similar to the way visual cortex recognizes objects where rubric is analogous to receptive fields.</p> <p>As neighboring rubric autocorrelates in striding displacements, DNN will discern to select the most likely resemblance and pass on to the next stack for further recognition.</p> <p>The net effect would be gross negligence of features to finest details. It does not really matter when it comes to recognition of the object as long as gross features distinctively recognizable e.g. the face of a tiger from monkey are distinct despite the loss of finely detailed features. However, for posture extraction, such gross negligence will introduce precision errors, such as labeling the left forelimb's ring finger from its internal reflection of the cylinder.</p>	Misconstruing features of interests, thereby introducing precision errors in whereabouts of the features of interest, especially in light of internal reflections in cylinder test.
ReLu (Activation Function) Errors	Nearly every layer of Resnet-50 employs a rectifier unit that attenuates negative outcomes on that layer. While this helps discerning specific features, higher order harmonics – manifested in non-linear noise – are introduced in posture extraction	Higher order harmonic noises in predicted posture coordinates due to uneven amplification of images
Picasso (Pooling) Errors	<p>DeepLabCut algorithm specifically searches for highlighted features of interest. It does not, however, associate whether these features make sense in where they are located. An example is the left paw must be attached to the left limb. These are often termed Picasso or pooling errors in DNN³⁶.</p> <p>In the present context, for example, an internal reflection of cylinder glass will project mouse forelimbs to a virtual image behind the physical cylinder (example as shown in Figure 17). DeepLabCut will recognize that virtual image as the forelimb if the actual forelimb was obstructed by the mouse body, for example.</p>	Misconstruing location of features of interest

Table 3: Sources of Error for ResNet-50, hence DeepLabCut

³⁶ A simplified discussion on DNN errors can be found in <https://towardsdatascience.com/what-is-wrong-with-convolutional-neural-networks-75c2ba8fbd6f>.

2.6 PalmGrid Signal Processing Module

Feeding successive images (signal) into ResNet-50 results in varying magnifications of signal samples depending on its signal amplitudes, producing harmonics noises to distort forelimbs localizations³⁷. Multiple signal processing filters are therefore required to sanitize the posture coordinates of these errors, prior to identification of wall rearing (Figure 18).

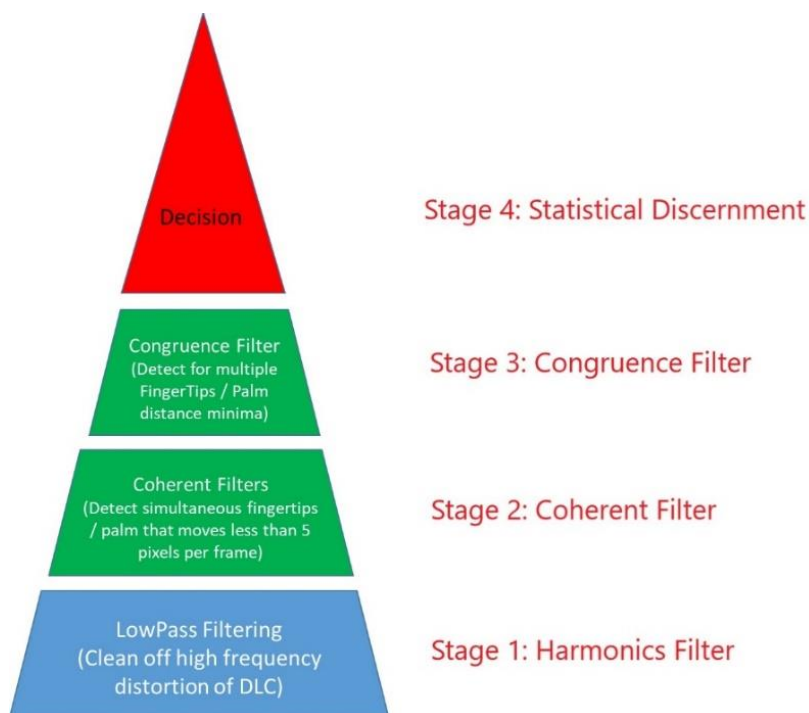


Figure 18: PalmGrid Signal Processing & Gauging Filters

2.6.1 Harmonics filtering

We have attempted several harmonics filters prior to identification of wall-rearing episodes. In prevalent image processing methods, OneEuro filter³⁸ with 1 Hz cutoff is often used to discern images from noises based on their differential (uncorrelated) statistical properties while maintaining sharp picture transitions on another. Moving average filter was also attempted that

³⁷ For discussions of activation functions and its relationship with harmonics noise, refer to <https://arxiv.org/abs/1603.00391>

³⁸ Introductory description of OneEuro filter is documented in <https://hal.inria.fr/hal-00670496/document>

track the prior and latter 0.5 seconds of predicted posture coordinates to predict the current coordinates. We would first like to test which filter provides effective performance to minimize harmonic noise sources introduced by rectifiers (ReLU) activators.

I have inspected what happened when predicted forelimb locations of DeepLabCut parses through both filters to compare their respective merits in our forelimb localization applications. This was done by overlaying the predicted forelimb locations to the mouse videos and visually observed their respective differences. An appropriate filtering algorithm should offer good proximities of predicted forelimb locations from the actual forelimb in the video image, with the predicted ones less jittery around the actual images.

It was observed that moving average filter was better than OneEuro filter, in line with what was believed. This is because errors introduced by ResNet-50 ReLU functions were not uncorrelated from what is to be predicted³⁹. Likewise, forelimb locations between nearby frames are strongly correlated as they move in a given trajectory. Therefore taking the average ± 0.5 seconds of particular frame in time offers less jittery predictions even if one frame was incorrectly localized.

On the other hand, the moving average filter does not eliminate Picasso and Max-pooling errors. If circumstances in the video causes a virtual image of a thumb to be sharper quality than the actual thumb, for example, ResNet-50 would still pinpoint the virtual image as the thumb – regardless whether that thumb is attached to the arm or detached somewhere else⁴⁰!

³⁹ For discussion of DNN algorithms and its relationship with associated noises, please refer to 35.

⁴⁰ For the very reason of Picasso and Max-pooling errors of ResNet-50, DeeperCut [31] performed extra recovery process of localization known as clustering and linear optimization. DeepLabCut [24] did not implement these processes.

2.6.2 Conversion to Polar Coordinates

Contingent on this sequence of wall-rearing events depicted in 2.1, conversion to polar coordinates helps making sense of predicted forelimb locations in cylinder geometry to inspect the synchrony of digits and palms as the forelimbs rear cylinder wall. Polar coordinate system is therefore utilized in gauging and discerning touches, by inspecting changes in radial displacements of successive fingertips. Identify coherent digits furthest away from the center of cylinder in polar coordinates with respect to cylinder center would be much easier in polar coordinate system than in cartesian coordinates with respect to camera field of view, as shown in Figure 19.

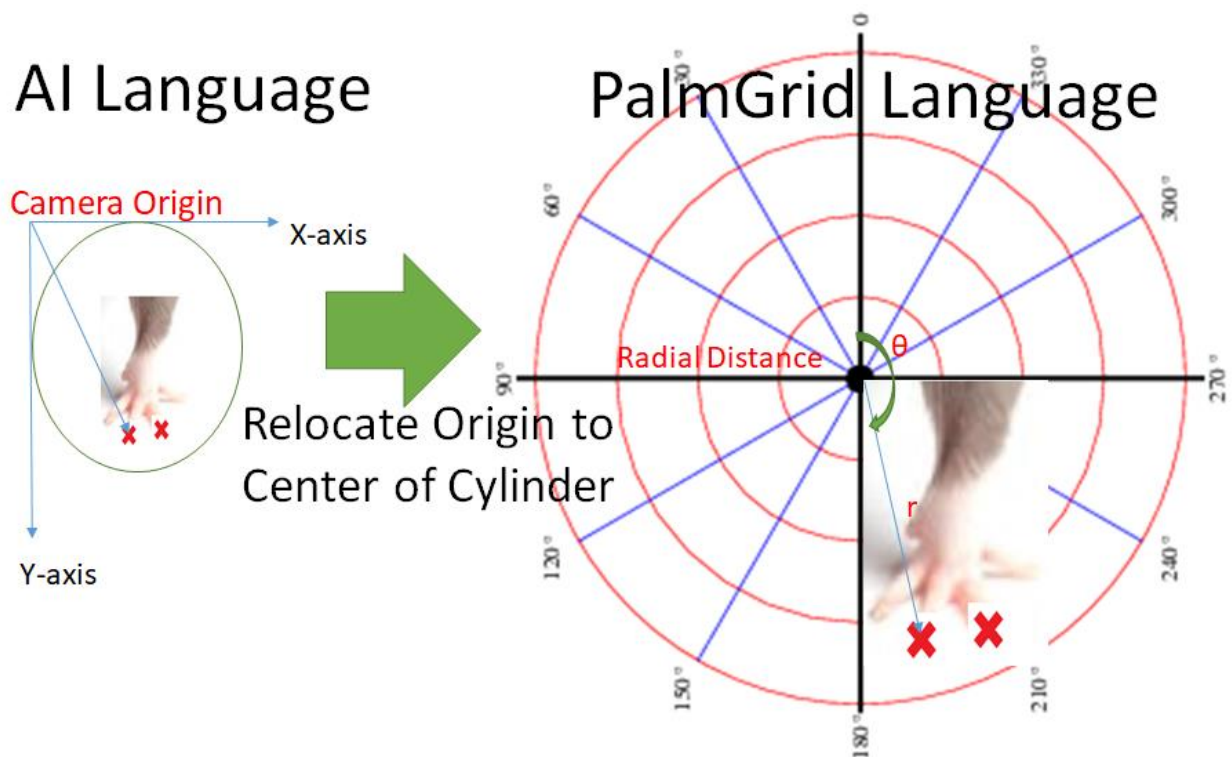


Figure 19: Conversion of DeepLabCut forelimb coordinates to PalmGrid language in polar coordinates help discerning synchronies in digits and palms

2.6.3 Extract Slow-Moving Frames

Wall rearing activities of the mouse are obstructed by the cylinder wall, resulting in changes in movement trajectory and/or momentary leaning towards the wall (an example illustrated in Figure 12). Because the mouse learned of such obstructions as it experienced, its paws will slow down somewhat when they approach the cylinder wall.

Leveraging the observation, gauging wall rearing activities shall start by inspecting individual fingertips coordinates that exhibits slowdown in digits displacements⁴¹ as the forelimbs approach cylinder wall, followed by its subsequent retracements. Slow-moving frames are thus extracted.

2.6.4 Extract Coherent Fragments

It is observed that the degree of freedom of mice digits is not as flexible as human or primates, in that their fingertips move in tandem with each other for specific movements. For example, its digits extend together as it moves forward to rear the wall. Likewise, their retracements are also in tandem with each other.

Leveraging the observation, we extract coherent fragments where a minimum of two successive fingertips slow down below 5 radial pixels per second (i.e. 0.1% of cylinder radius) in the same direction at the same time, until the point when the digits accelerated and retraced from cylinder wall. We also examine the direction of the digits' movements within 0.5 seconds: a wall-rearing touch is likely happening if the digits radial distance is moving towards the wall prior to its slowdown, while subsequently retracing back after the rearing.

⁴¹ In our default template slow moving frames are defined to be those whose fingertips' radial distances that move less than 5 pixels in adjacent image frames for a given fingertip. This is 0.1% of the cylinder's diameter.

The time fragments that correspond to simultaneous digits following the sequence of events as approach, slowdown, rearing and finally retracement of fingertips is termed **Coherent Fragments**.

2.6.5 Identify Congruence Points

It was also observed that when the palm hits the wall, there is an alteration of paws' movement trajectory. Since the digits are hit and obstructed by the wall first whilst the momentum of the palm continues, we observe digits-to-palm distance hitting minima followed by a change in trajectory:

The instances in which these minima occurred are termed **Congruence Points**. It is also observed that successive digits-palm minima of a forelimb do not necessarily happen at the same time, but very close to each other – usually within *one* second.

2.6.6 Remove Outliers of Refined SubFragments

If predicted forelimb coordinates met the two congruence filters criteria with nearby congruence point(s). Its forelimb(s) is 1) slow moving, 2) moving towards the cylinder wall and then retrace, and 3) altering its movement trajectory. I named fragments that met these criteria as **Refined SubFragments**. Questions remain whether such refined subfragments are wall rearing, paws idling in free space or slow crawling on floor (see these postures in Figure 20). Outlier removal filters are used to eliminate identified wall-rearing that did not make sense.




		
Wall Rearing	<p>Slow Crawling</p> <p>Mouse slowly hoovering and encircling cylinder floor. With ResNet-50 jittery predictions of forelimb digits, some of these episodes will meet both coherences and at the same time changes in trajectory</p>	<p>Paw hanging</p> <p>Mouse lifted the forelimb and stay there with “stand up fists” while its body leaning forward and backwards. With ResNet-50 jittery predictions of forelimb locations, digits will be in slow movement, forward and retrace, as well as congruence</p>

Figure 20: Some postures that fulfill two coherences and congruence criteria

Statistics of these Refined Subfragments are calculated to identify wall-rearing activities from other floor maneuvering activities. For each episode, we observe averages and standard deviations in radial displacements of digits and palms. I found that forelimbs cannot be rearing if they are resting on the floor or crawling very slowly, implying the standard deviation of radial displacements within these identified episodes are very small (within 5 pixels, or 1% of cylinder diameter). Likewise, the forelimbs cannot be rearing the wall with localized digits lying within parallax of inner radius as shown in Figure 21.



Figure 21: Removal of detected wall-rearing episodes where majority of digits lie within parallax of inner diameter

These outlier removal steps are detailed as follows:

1. For each Refined SubFragments, the mean and standard deviations of the five digits, as well as the palm location, are computed;
2. To segregate floor resting or crawling episodes from its wall-rearing counterparts, it was noted that variations (standard deviation) of its fingertips within the refined subfragment is lower than 5 pixels (or 1% of cylinder diameter). If a number of digits coincidentally exhibit such slow movement traits, the episode is treated as floor resting episode and removed.
3. If all fingertips and palm are all detected to lie within the inner diameter of cylinder observed through the parallax effects from the bottom (Figure 21), it is unlikely that being a wall-rearing activity. Outliers whose majority of digits lie within inner diameter

are also removed. These inner diameter measurements are measured during experimental setup stage, using Microsoft Photos or ImageJ to measure the number of pixels it spans in camera field of view.

2.6.8 Gauge Refined SubFragments into wall rearing episodes

At the end of all these filtering, refinement, signal processing and outlier removals, final sets of wall-rearing sub-fragments are established. These subsets of refined subfragments are then consolidated into a wall-rearing episodes that facilitate qualitative assessment of wall-rearing in later stage. The wall-rearing episodes are tabulated in Wall Rearing episodic report.

2.6.9 Export of wall-rearing episode

The final wall-rearing episodic report is thus compiled. A sample report format is presented in Figure 22. In the report, each episode is given its midpoint, start and end frame reference that facilitates traceability of wall-rearing episodes in numeric report. An example is given as underlined in Figure 22, where the algorithm notes that between frames 364 to 760 there is at least one digit of the forelimb moving slowly. Within this slow-moving fragment, two wall-rearing episodes between frames 498 to 528 and frames 611 to 663 are respectively identified. Each of these wall-rearing episodes represents at least two digits of slow movements, while changes in movement trajectory occurred. The forelimb was also noted of approaching and retracing from wall.

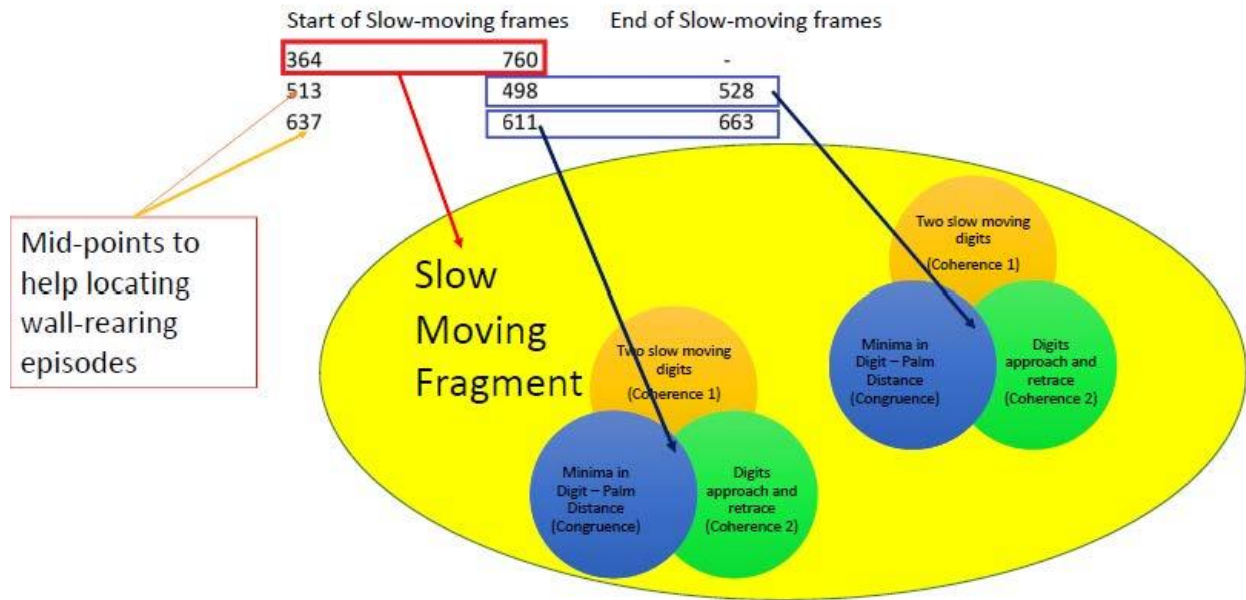


Figure 22: Constituents of Wall-Rearing Episode (Numeric) Report, which tabulates all evaluated wall-rearing episode in terms of Slow Moving Fragments; and episodes that met 2 coherence and congruent criteria with outliers removed.

2.6.10 Compile wall-rearing episodes into video fragments

As noted in Chapter 1.2, qualitative assessment of wall-rearing enhances the outcomes of cylinder test protocol, by having investigators reviewing respective synchronies of forelimbs as mice rear the wall. This is especially relevant in stroke and recovery assessment applications, where mice subtly avoid explicit wall sliding activities by having the other normal limb pushing against the wall for dismounting in a scenario termed “paw-dragging”.

The following figures extracted from [30] readily demonstrates the paw-dragging scenario where one side of its motor cortex was given focal ischemic stroke. As the mouse stands on its rear legs to explore the cylinder wall then drags, its affected (contra-lesional) paw drags along the cylinder wall towards its midline or down the wall; while its unaffected forepaw provides postural support against the wall (Figure 23). Prior to the contra-lesional forelimb detaching

from cylinder wall, unaffected forelimb will assist the dismount to land on cylinder floor. This explains why if one simply “listens” to timing of touches of forelimbs on cylinder wall, it is very difficult to discern progress of stroke and recovery.

Paw-drags rarely occur in uninjured mice. Therefore appearance of a paw-drag is a positive indicator of injury to the forelimb sensorimotor cortex.

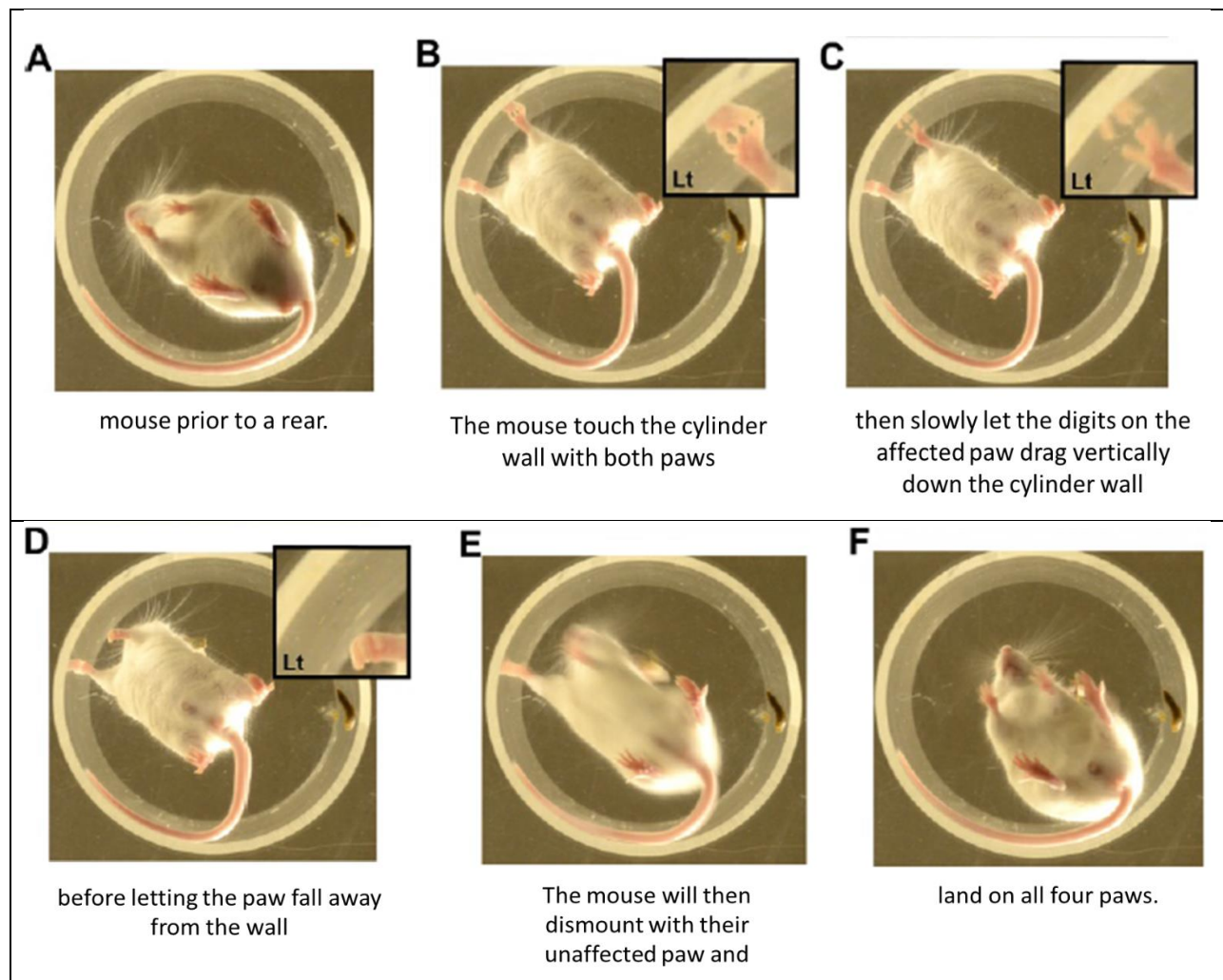


Figure 23: Synchronies of forelimbs in wall-rearing when focal ischemic stroke was given to one side of sensorimotor cortex ET-1. Images obtained from [30] Roome & Vanderluit. Paw-dragging: a novel, sensitive analysis of the mouse cylinder test. *J. Vis. Exp.* (98) (2015)

Since we evaluated respective frames of wall-rearing episodes, we make use of these identified episodes to segment activity videos recorded into smaller videos focused in wall-

rearing using off-the-shelf multimedia conversion software FFmpeg. In doing so, investigators can focus to evaluate individual quality of rearing, as well as quickly discern if there are false rearing that could be missed.

2.7 Experimental Testing

Testing of PalmGrid algorithm was conducted with a random sample of six healthy male mice (n=6) from different cohorts randomly chosen of age between 2 to 6 months. We used EMX-1 mice⁴² strain as the subjects to conduct experimental testing. All animal procedures were approved by the University of British Columbia Animal Care Committee and conformed to the Canadian Council on Animal Care and Use guidelines. As movements of healthy mice shall not differ in gender, the randomly chosen experimental cohorts were male.

For each mouse, three separate videos (each lasting around 3~4 minutes in compliance to cylinder test existing protocols) are recorded in random times and then analyzed in PalmGrid station. As reported in Section 2.4.1, a different mouse strain AI-94 was used to train the PalmGrid station, so as to test our hypothesis that features knowledge can be learned independently of the phenotype. Focal settings of the camera were not altered from one videotaping to another.

The training of PalmGrid station using AI-94 recording took 7 hours in excess of 200,000 iterations using Nvidia Quadro 5GB Graphics Processing Unit described in Section 2.2.2, as recommended in DeeperCut [31]. Analysis of each subject video took about 20 minutes, together with another 15 minutes to convert the posture extracted images back into a video clip

⁴² Jackson Laboratory Emx1-IRES-Cre mice; Stock No: 005628 | Emx1IRES cre

in labeled fingertips and palm (Figure 24). Both training and conversions are done by batch and unattended, relieving the laborious manual monitoring that the exercise was set out to save in the start.

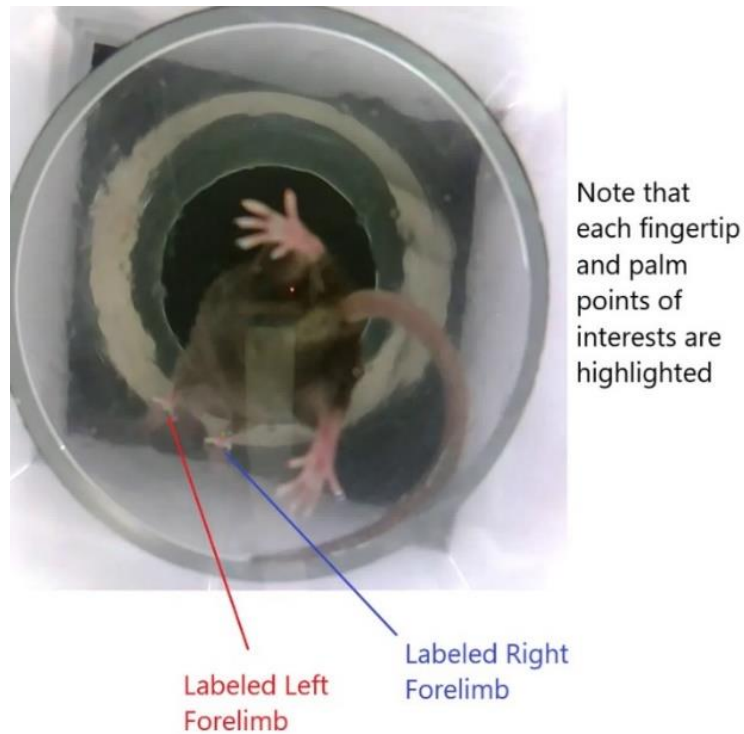


Figure 24: Sampled Labeled Images

More samples and details of training video labeling are found in Section 2.4.1 and Appendix F respectively.

2.8 Results

To evaluate robustness of PalmGrid process, the Wall Rearing Episode (Numeric) Report was compared with the results of the Ground Truth report. Methods and results to compile ground truth reports are documented in Appendix G. The results from ground truth raters agreed with each other.

Five mice were chosen out of the six sample subjects, with two video clips of each are being randomly chosen and analyzed. Each wall-rearing episode was then compared with corresponding ground truth rearing to assess the accuracies of algorithms. An example of the Wall-Rearing report, being the output of the PalmGrid process, was depicted below (Figure 25) that refers to the totality of left and right paws detection.

2.8.1 Overall Results

Figure 25 tabulates the overall results. Appendix C.2 tabulates the corresponding overall results in percentage terms for each video frame. In the report, detected wall-rearings by PalmGrid either positively reflected actual wall rearing (termed “Correctly” in Figure 25), or falsely misconstrued the activities as rearing (correspondingly termed “False Positive”). Either a detected touch by the PalmGrid process reflected correctly the actual wall-rearing touches, or it reported false alarms where wall-rearing was detected by PalmGrid algorithm but not found in Ground Truth Report. The percentage of “Correctly” and “False Positive” touches added up to 100% of total touches in each video frame.

There were also False Negative (missing) episodes that reflected actual touches on the cylinder wall that are not detected by the PalmGrid Process. This was expressed as a percentage of the total touches detected by the Palmgrid algorithm.

Percentage of correct detection was calculated by actual wall-rearing hits divided by the number of detected hits; while false positives refer to those detected episodes with no actual hits. Finally, false negatives refer to actual wall rearing that PalmGrid omitted. All of these are expressed both in actual numbers and in percentage as depicted in Appendix C.

Correct recognition rate on 10 video streams – with 1 left and right forelimbs videos for each cohort – came to 70%. In terms of numbers, a total of 406 wall-rearing touches were recognized from the manual observations of 474 wall-rearing touches, implying $406 \div 474 = 0.856$ *or* 85.6% of wall-rearing episodes were detected in aggregate when all false positives were taken away.

Of all the 580 detected wall-rearing episodes of PalmGrid, false positives (alarms) percentages came to $174 \div (406 + 174) = 0.3$ *or* 30% Meanwhile, false negatives came to $68 \div (406 + 68) = 0.1172$ *or* 11.72%.

		Actual	Congruence Filter Performance			Outside CZ
			Inside CZ			
		Touches	Correctly	False +ve	False -ve	
EMX01-01	Left	27	18	9	4	
	Right	23	14	9	4	
EMX01-02	Left	15	11	4	1	
	Right	11	5	6	8	
EMX02-01	Left	16	13	3	1	
	Right	18	8	10	5	
EMX02-03	Left	16	8	8	2	
	Right	11	4	7	5	
EMX03-02	Left	52	41	11	1	
	Right	53	40	13	3	
EMX03-03	Left	45	36	9	1	
	Right	43	35	8	5	
EMX04-02	Left	32	23	9		
	Right	29	19	10	5	
EMX04-03	Left	42	32	10	2	
	Right	51	34	17	1	
EMX06-01	Left	45	33	12		
	Right	26	20	6	14	
EMX06-02	Left	9	6	3	2	
	Right	16	6	10	4	
Total		580	406	174	68	0
Success Rate%			70.0%			
False Positives%				90%		
Undetected %					12%	

Correct	Algorithmic rules detected there should be at least one Touch within the slow moving fragment; and actually there are
False +ve	Algorithmic rules detected that there should be at least one Touch within slow moving fragment; but actually No Touch
False -ve	Algorithm did not detect anything within the slow moving fragment; but actually there was at least one Touch
Outside CZ	Actual Touch that lies beyond Congruence Zone Detections; usually due to obstruction, start of video frame etc.
Success Rate %	Total Number of Actual Wall Rearing / Number of detected Hits
False Positive %	Total Number of False Positives / Number of detected Hits
Undetected %	Total Number of Undetected Hits / Total Number of detected Hits

Figure 25: Test Results of PalmGrid for a cohort of n=5

2.8.2 Independent Assessment of Left and Right Forelimbs

As PalmGrid allows independent assessment of left and right forelimbs, the corresponding touches and its percentage were separately analyzed. Results of such analysis are depicted in Appendix C.3 to C.6.

For the right forelimb (Appendix C.4), mean correct recognition rate comes to 68.1% inclusive of the outliers. False Positives came to 31.9% whereas false negatives came to 18%. For the left forelimb (Appendix C.6), two video recordings of EMX-02 cohort exhibit outliers, and the correct recognition rates come in between 67% to 81% beside the outliers. Mean correct recognition rate comes to 73.9% inclusive of the outliers. False positives came to 26.1% whereas false negatives came to 5%.

2.9 Discussion

2.9.1 Correctly Recognized Touches

To the knowledge of the author, this is the first attempt by which artificial intelligence algorithm is used in precision applications to extract posture in the context of the cylinder test. ResNet-50 used in this process achieved 5.25% Top-5 recognition error in ImageNet competition of 2015. We learned that Top-5 Classification error – being the prevalent benchmark of DNN algorithms – is not equivalent to Precision (or Top-1 localization) error. A similar study of precision error of ResNet with 50 and 101 layers of depth was given in [21] that depicted respective 76% and 78% localization (Top-1) accuracies. The PalmGrid algorithm anticipates inherent inaccuracies of ResNet-50 to identify wall-rearing episodes.

With roughly 1 in every 4 errors in forelimbs' location predictions in errors, identifying 70% correct recognition for PalmGrid Process is a good initial attempt using basic laboratory equipment. The process may improve from this rate simply by reducing acrylic reflections through anti-acrylic paints; or by organic improvements in future artificial intelligence algorithms. As DeepLabCut came in as open-source, future adoption in more advanced DNN/posture extraction algorithms should enhance localization accuracies of PalmGrid⁴³.

2.9.2 False Alarms (Positives) and Error Propagation Modeling

False Positives in the algorithm refers to the fact that the algorithm believes some wall-rearing touches had occurred in the highlighted episode but in fact there was none. The high

⁴³ DeepLabCut algorithm is derived from the corresponding human posture extraction algorithm DeeperCut, that has three components: AI recognition, Clustering, and Linear Transformation using around 2,000 photos in training. The software cut the clustering and linear transformation part to reduce overheads in computations and learning. If DeeperCut is used, it is tested that recognition rate will rise from 76% to 82% [31].

false positive rates (30%) were largely attributed to ceiling precisions of prevalent Artificial Intelligence Algorithms. For the ResNet-50 chosen, precision – defined as Top-1 Classification – has its mean ceiling accuracies of 76% (Figure 13).

As a rough estimate to account for the false positives rate, since ResNet-50 gives 76% (Top-1) localization accuracy, this implies probability of error in extracted postures of each digit is $(100 - 76) \% = 24\%$. These errors are then propagated into the signal processing filters in Figure 18, reproduced below in Figure 26 with the error propagation model.

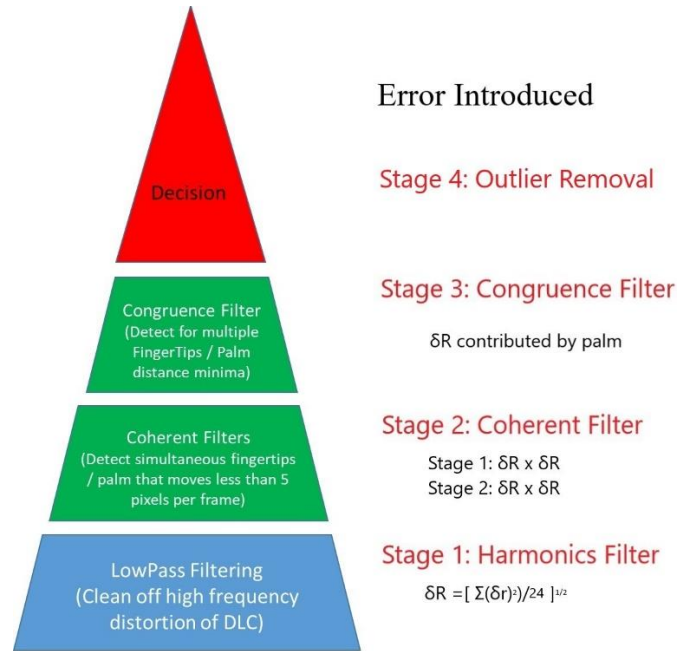


Figure 26: PalmGrid Error Propagation Model

In this rough estimation model, we assume digits and forelimbs are independently predicted by ResNet-50. This assumption is justified as we inspected DeepLabCut [24] open source code and see little evidence of the algorithm using posture recovery methods such as clustering and linear optimization as its predecessor DeeperCut [31].

The first stage of PalmGrid signal processing filter averages 24 signal samples: 12 from priors and 12 from corresponding latter time samples. These imply an input error rate of $(0.24^{2/24})^{1/2} = 0.0489$ or 4.89%, equivalent to an accuracy probability of $(1 - 0.0489) = 0.951$ or 95.1%.

The Coherent Fragment extraction introduces two stages of errors. First (Stage 1) it requires a minimum of 2 digits to be slow moving, implying probability of accurate Coherent detection to be $0.951 \times 0.951 = 0.9044$ or 90.44%. Next (Stage 2) the slow movement fragments will propagate to discern of forward to and retracements from the cylinder wall. Similar to the above modeling, the filter cascades another factor of 0.9044 (90.44%) to the propagated errors.

The congruent point detection will then assess palm posture closest to the fingertips some point within a coherent subfragment. This will introduce another 0.951 (95.1%) factor in the probability of success. Statistical filters will not introduce significant errors, as they are concerned only with specific criteria, rather than mathematical calculations.

As we assume the coherences and congruence to be independent activities, these three core filters result in a maximum accuracy of detection of $0.9044 \times 0.9044 \times 0.951 = 0.7778$ or 77.8%⁴⁴. Therefore 70% actual recognition by PalmGrid is in line with the basic setup that provides rough wall-rearing estimates in cylinder test.

We also note that because the mice under experimentation are freely moving, the chance of obstructing the camera by body is higher. This explains differential mean recognition rates of left and right paws at 73.2% and 68.1% respectively. After all, Neural Networks is a statistical-based algorithm that will discern slightly varying wall-rearing decisions. And because of these obstructions and internal reflections of glass cylinders, as opposed to head-fixed mice in

⁴⁴ If coherence and congruence are not independent, the maximum accuracy will simply decrease as we have to account for joint probabilities between activities.

otherwise experimentations, we noted that outliers in detection rates readily exist. In our case, cohort EMX02 in her first recording exhibit significant outliers. In her second recording, the left forelimb demonstrates significant outlier. These bodily obstructions of digits significantly hindered recognitions, hence reduce correct recognition rates. The other major reason behind these false alarms was the lack of depth cue. With a single camera, it is difficult to discern between the forelimbs rearing the wall or crawling on the floor plane that corresponds to the wall-rearing projections (Figure 27). Resolving the issue requires installation of depth camera or relying on future enhancement of Pi-camera with depth resolution features. At the time of writing, these measures imply extra costs of procuring specialty types camera.



Did mouse rear the wall or forelimb on floor?

Figure 27: Lack of depth cue made resolution of wall-rearing challenging from corresponding floor rearing around the same projected diameter

2.9.3 False Negatives or Missing detections

Meanwhile, there were omissions (false negatives) that PalmGrid did not recognize of plausible wall-rearing that has factually occurred. A 12% overall false negatives rate is in line with error propagation model, demonstrating the encompassing nature of the algorithm in discerning potential touches. Any refinements in anti-acrylic paints or enhanced algorithms will reduce posture extraction errors, improving the false negative rates.

The major reason behind the omission was DeepLabCut located the wrong palm coordinates as its features blurred. While features of digits of forelimbs are easily discernable, features of the palm poses challenges to ResNet-50. This is especially true when the images are compounded with confounds such as slow shutter speed, internal acrylic cylinder reflection, refraction due to mouse urinations or when forelimbs were raised above certain heights that basic Pi-camera fails to discern palm features. Since DeepLabCut simplified DeeperCut [31] features that recovers localization conflicts (such as use of clustering and linear-optimization algorithms), its recovery capability of less discernable features (such as palm under fast movements) will be weak especially in moving subjects. In an analogy, human cortex will possess difficulties to resolve images if the eyes are suffering from diseases. Given the limitations of basic Pi-camera, the 12% omission rate shall be anticipated.

To further improve false negatives, a faster shutter speed Pi-camera with wider focal range is recommended, that allows freely moving mouse to be captured in higher precisions and resolutions of its forelimbs. Anti-acrylic reflection paints will somewhat help to avoid ResNet mistakenly recognize the virtual image of forelimbs as the real one. DeepLabCut shall also be enhanced with clustering and linear optimization, that despite poor resolutions the palm locations can be somewhat recovered. On hindsight, a combination of the above will help.

All above improvement measures require procurement of advanced camera or substantial enhancement of DeepLabCut. Both were not intended as we set out our research objectives in terms of costs and sophistications. To procure advanced camera certainly imply higher costs of adaptation, while enhancing DeepLabCut with clustering and linear optimization will require 2,000 posture photos to be prepared in training phase [31] together with an advanced GPU to process the sophisticated posture extraction algorithms. It is worthwhile to proceed further research in these directions.

2.9.4 Different correct recognition and omission rates for left and right paws

It was also noted that significant differences exist between left and right forelimbs' correct recognition and omission rates. A summary of recognition rates based on results depicted in Appendix C for the experimental testing is given in Table 4.

	Correct	False Alarms (false positives)	Omissions (False Negative)
All left forelimb	221 (73.9%)	78 (26%)	14 (5%)
All right forelimb	185 (68.1%)	96 (32%)	54 (18%)

Table 4: Computed Wall-Rearing Results of Left and Right forelimbs

From prima facie evidence shown above, it seems trivial to conclude that biodiversity differences of left and right forelimbs were evident from these results. However, such conclusion may be premature given the shutter speed, focal range, and lack of depth cues limitations of existing camera capabilities. I would therefore leave the preferential inference to later stage.

2.9.5 Making use of Wall-Rearing Episode Report to enhance efficiency

Current protocol of Cylinder Test favors investigators to review the wall-rearing fragments to assign their own scores according to their specific concerns. To address the requirements, PalmGrid compiles identified wall-rearing episodes into a video as an example to demonstrate efficiency enhancements.

In a typical 4 minutes recording, 6,000 frames⁴⁵ have to be reviewed in manual labor approach. Even if the investigator reviewed each image with 1 second per image, such review will take around 6,000 seconds (~1.67 hours). As a motivating example to make use of the wall-rearing episode report, a sample video recording of EMX03-02 is used. PalmGrid extracts 26 wall-rearing episodes from the extracted postures, followed by compiling the wall-rearing fragments into respective smaller video recordings using FFmpeg. It is noted that in this particular case, the smaller video fragments shortened review time to 275 seconds (4.58 minutes) after removal of non-rearing time fragments, implying significant reduction in review efforts at marginal costs. Actual time saving varies, depending on how many vertical explorations the mouse use in a given recording and the fluencies of investigators to use different tools in conducting wall-rearing reviews.

On that note since wall-rearing episodes are now automatically extracted and detected, there is no reason why the Cylinder Test observation period cannot extend beyond 10 minutes. Time savings achieved using PalmGrid demonstrates further advantages compared to conventional frame-by-frame review approach.

⁴⁵ 4 minutes x 60 seconds x 25 frames per second in a given video recording = 6,000 frames

2.9.6 Comparison of labor time required to use PalmGrid

Are there time savings to utilize PalmGrid algorithm? To evaluate if there are efficiency gained, I compared time taken to use PalmGrid versus the method described in [30] to identify wall-rearing, the video fragment EMX03-02 is fed into PalmGrid to estimate incremental labor hours required versus corresponding manual labor hours to review in paw-dragging method.

In PalmGrid the video is fed into an automated pipeline to localize forelimbs and identify wall-rearing episodes; whereas two iterations to slow play the video is required in method described in [30] to skim through and verify all rearing. Because wall-rearings in PalmGrid are compiled into smaller video fragments, reviewers can focus on each smaller fragment to conduct qualitative analysis that reduces possibility of errors in qualitative assessments. Frame by frame qualitative review can be avoided as reviewers can conduct the qualitative review in different times due to smaller video fragments, that poses smaller chances of laborious fatigues and errors.

As a benchmark comparison, my take of time comparisons using EMX03-02 is tabulated below in Figure 28 using off-the-shelf video player VLC Media Player⁴⁶. In this sample comparison, smaller video fragments help reviewers to review shorter video fragments in 0.6x actual video speed, rather than slowing down the reviews to more careful 0.25x to avoid omissions. There were no timing overheads in noting down the wall-rearing episodes as analysis and identification of vertical explorations are machine driven, leaving qualitative assessment phase to investigators in higher quality. My illustrative comparison came to around 92% time

⁴⁶ VLC Media Player is a general public use, off-the-shelf video player downloaded from <https://www.videolan.org/vlc/index.html>

saving. Actual savings shall vary for different investigators employing other protocols.

Tasks to analyze stack of 4 min videos	PalmGrid	Manual Based on Roome & Vanderluit (2015)
Benchmark on a 4-minutes Video (EMX03-02) using VLC Player		
Assessment Method	<ul style="list-style-type: none"> Machine extracted wall-rearing episodic videos For each episodic video, slow play in 0.6x speed and label whether it was rearing / paw dragging 	<ul style="list-style-type: none"> Play video in 0.25x speed Human rater mark down all rearing; Repeat once to verify For each rearing, review frame-by-frame to label rearing / paw dragging
Benchmark Time for review	26 videos extracted total 2.75 minutes under 0.6x slow play = 2.75 / 0.6 = 4.58 minutes	1. Slow Play Phase = 4 minutes / 0.25 = 16 minutes plus 35% overhead 2. Reconfirmation = 16 minutes plus 15% overhead 3. Frame by frame review of all marked down rearing frames = 1,104 frames
Total Time	4.58 min x 60 seconds = 275 seconds	16 min x 60 sec x 1.35 + 16 min x 60 sec x 1.15 + 1,104 frames x 1 sec = 3,504 sec
Efficiency Achieved	~ [1 - 275 / 3504 = 92.1%]	

Figure 28: Illustrative comparison of PalmGrid assessment time versus methods described in Roome & Vanderluit [30]

2.9.6 Benefits of the software approach

There are many benefits of such achievements, in which 1) it relieves significant laborious tasks that were only achievable through prone-to-error human observations, and cumbersome post-experimentation data processing; 2) it allows independent assessment of left and right forelimb movements in ischemic strokes and its corresponding rehabilitation; 3) it does not require complex calibration before experimentation such as multiple camera synchronizations. Last, not least, it requires minimal setup costs with a basic Raspberry Pi video camera system and simple laboratory equipment. These lowered entry cost and skillset challenges are set to benefit laboratories in their use of precious resources.

Chapter 3 Design Choices and Discussions

3.1 Strength of PalmGrid

The obvious strength of PalmGrid stems from its software-driven simplicity even without meticulous hardware calibrations. In the conventional approach conducting cylinder test, precious manual labor has to be exploited to count wall-rearing activities which are laborious. It is also burdensome as reviewers are required to skim through many videos, and in worst case frame-by-frame. To maintain acceptable level of accuracies, methods adopted in [25-27, 30] often requires three parses for each video review per investigator. As an illustration extracted from methods depicted in [30], first parse requires raters to skim through video in 0.25x speed to write down respective wall-rearing of forelimbs, followed by a second parse to verify. To review quality of individual rearing requires, in worst case, frame by frame review to note of paw-dragging. Overheads readily exist in each of these parses, and video pauses, rewinds, and fast forwards are often required that staggered up overheads in the review.

The use of capacitive touch sensors somewhat helps by detecting touches, but it fails to differentiate which limb touched the grid as well as the quality of rearings, despite extra costs involved in setting up electronic grid system. Study of contralateral stroke impacts to movements using capacitive touch sensors approach proved challenging to assess quantity and quality of wall-rearing.

PalmGrid uses minimally visible lights to discern wall-rearing touches. As long as light intensity enables artificial intelligence algorithms to recognize forelimbs, its setup cost is otherwise minimal. Complex setup procedures and sophisticated calibrations are not required in PalmGrid, as opposed to touch sensors where careful planning of sensor grid will be required. In

our experiment that assumes little technical competence in basic laboratory equipment, our testing shows that 70% of wall-rearing are identified. It is therefore a competent tool for pre-selection of wall-rearing, leaving investigators' precious time to assess quality of these rearings, and their specific features-of-interests such as wall-sliding and wall-dragging.

We have deliberately avoided more sophisticated equipment (such as depth camera, multiple cameras, and anti-reflective paints) and meticulous fine-tuning to enhance detection outcomes. Based on inspection of false alarms and omissions, use of depth camera or multi-camera approaches to resolve depth and focus should enhance accuracies, at the expense of increasing costs and complexities. High-resolution Pi camera was not adopted, as we set out to evaluate what basic equipment could do. Indeed, 70% accuracy is not bad given all these constraints without any fine tuning⁴⁷, demonstrating the extent of artificial intelligence can assist in neuroscientific investigations. Any measures to fine tune the configurations will bridge prevalent accuracy gap from 70% to theoretical maximum of 77.78%⁴⁸.

The outcome of PalmGrid extracts wall-rearing episodes in report as well as wall-rearing videos. This will dramatically save time to conduct the experiment without laborious analysis to focus precious time resources on higher quality tasks to assess details of wall-rearing episodes. In our testing of a four-minutes video with 26 wall-rearing, conventional protocol to review 6,000 video frames is now "shrink-wrapped" into 275 seconds (4.58 minutes) of 26 videos that achieves 92% time saving. If we leverage the feature to round-the-clock monitoring of stroke mouse where wall-rearing occurrences are less frequent, significant analytical time savings can be achieved by eliminating reviews of non-wall-rearing postures. In a similar manner, we can

⁴⁷ A typical high-resolution camera of Nikon, such as M12 lens, will add up another few hundred dollars in costs

⁴⁸ For discussions of error propagation model, please refer to Section 2.9.2.

increase the sample size of test subjects to analysis of bigger rodent samples; since artificial intelligent machines are now tasked to perform most of the laborious pre-selections. The final wall-rearing videos offer flexibility for investigation of specific posture-of-interests in their own protocol such as wall-dragging or wall-sliding.

Given the automatic posture extraction and wall-rearing detection capabilities, use of PalmGrid algorithm can help to pre-select mouse of specific forelimb preferences in different circumstances. Similar to the cylinder test arrangement, one can readily engage bigger sample size to analyze their activities round-the-clock, extending its applications beyond cylinder test.

3.2 Limitations of PalmGrid

The obvious limitation of the PalmGrid experimentation setting is its requirements⁴⁹ of minimal lights to discern of movement and features. If specific protocol requires very dim lights ambiance, use of PalmGrid may not be suitable. In those cases, use of specific infrared camera might be an option worth trying.

Analysis of PalmGrid data requires powerful computers with efficient Graphics Processing Units (GPUs) that used to be a challenging end means for most laboratories. This barrier is gradually overcome with many universities offering GPU shared services and free Matlab / Python licenses that processes resources hungry artificial intelligence algorithms.

⁴⁹ For example if mice behavior study under dim infrared lights is required.

3.3 Future Improvement Areas

3.3.1 Use of more advanced artificial intelligence algorithms

There are many rooms for future enhancements, and 70% recognition rate can at best taken as an encouraging milestone. As DeepLabCut enhances through porting to more advanced artificial intelligence algorithms such as Inception version 4.0⁵⁰ (Figure 13), it is expected that continuous precision enhancement beyond 76% will, in turn, advance DeepLabCut posture extraction accuracy. For the sake of arguments using the rough estimate of error probability in the Discussion section, expected PalmGrid accuracy can advance to $0.92 \times 0.92 \times 0.96 = 0.81$ or 81% if we change the core algorithm from ResNet-50 (76% Top-1 accuracy) to Inception v4 (80% Top-1 accuracy). As graphics processing unit prices come down further, choosing more resources hungry graphics processing unit will help to advance PalmGrid performances.

3.3.2 Use of more advanced posture extraction algorithms

DeepLabCut is derived from more sophisticated human posture extraction algorithm called DeeperCut [31] that further recovers errors of ResNet algorithms with clustering and linear optimizations. These steps require higher power Graphical Processing Units while advances posture extraction accuracies from 76 to 82% tested on ResNet-50 algorithm. The 6% enhancement comes with extra costs in preparing 2,000 images training the artificial intelligent machine. Using the same formula given in Section 2.9.2, ceiling accuracy will increase by 6%. It is therefore expected corresponding increase in wall-rearing episodic extractions accuracies.

⁵⁰ As reported in [21], Inception version 4 has 80% accuracy with 12G operations in one forward pass.

3.3.2 Dual-camera, epipolar geometry approach

Further experimentations based on two cameras approach using epipolar geometry⁵¹ shall be another worthwhile attempt, in that the depth of posture can be derived from simultaneous observations of two cameras. Changing the approach from statistical gauging to a deterministic examination in the final outlier removal stage will further narrow the margin of detection accuracies from prevalent 70%. But it should be noted that there are added complexities in equipment calibration and increased costs in equipment provisioning.

New methods are now emerging that allows Raspberry Pi to record with two video cameras⁵² in synchrony. This bypasses the synchronization constraints between camera recordings, hence enabling the use of epipolar geometry that changes latter part of PalmGrid gauging algorithms from statistical estimation approach (in steps laid out in Sections 2.5.7 and 2.5.8) to deterministic approach where actual coordinates of fingertips are computed.

While we are on the topics of multiple cameras, one can hypothetically increase the number of cameras beyond two in the hope that more perspectives of video recordings might overcome obstructions of features of interests to increase tracking accuracies. This may be a worthwhile option to consider when Raspberry Pi can load more than two cameras in synchrony. But as at our knowledge of Raspberry Pi to date, synchronizing Raspberry Pi video recording beyond 2 cameras is exceedingly cumbersome⁵³. In fact, if the multiple camera synchronization issues can be resolved in Raspberry Pi in its future versions, it is possible to generalize PalmGrid usage

⁵¹ For modeling of deterministic approach such as epipolar geometry, refers to textbooks that describe its methods. An example of such text includes Xu & Xiang (2013) Epipolar Geometry in Stereo, Motion and Object Recognition; Springer Science and Business Media ISBN: 9789401586689.

⁵² An example is Raspberry Pi 3 Model B with multiple camera adapter modules

⁵³ Refer to <https://www.raspberrypi.org/forums/viewtopic.php?t=212013> for prevalent discussions of multiple camera synchronization in Raspberry Pi

beyond cylinder test to generalized cartesian coordinates where exact paw locations can be tracked using multiple cameras.

3.3.3 Changing PalmGrid signal processing approach to machine learning

At first sight, the current market hype of superiority in Artificial Intelligence might induce ideas that post-extraction wall-rearing detections shall advance by cascading another machine learning system. Personally, I have some doubts. This is because according to Figure 13, the best machine learning algorithms achieve around 82% precisions at best. If we cascade another machine learning context to predict wall-rearing detections, the precisions will hit accuracy ceilings of $76\% \times 82\% = 62.32\%$, which is substandard to present achievements! One shall not rule out if future machine learning precisions increase beyond 90% Top-1 accuracy, this may well be a worthwhile attempt.

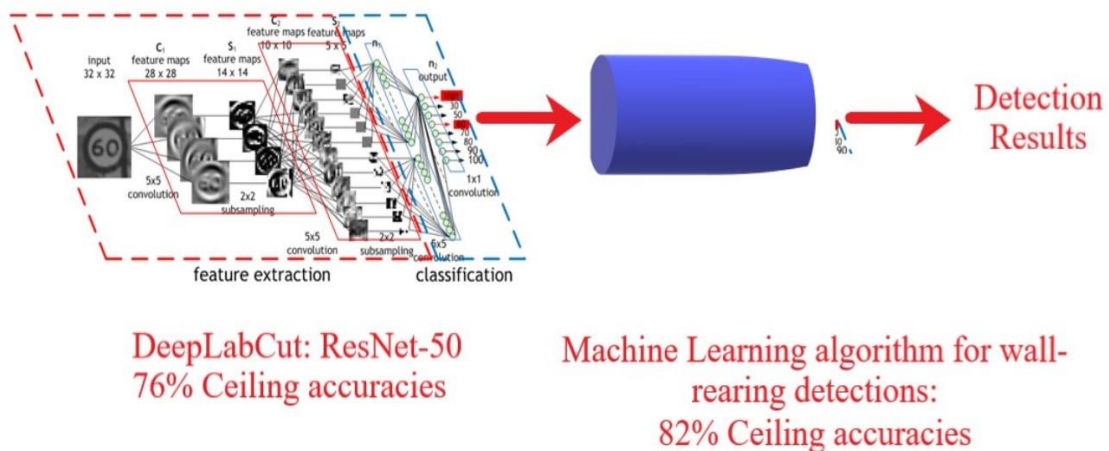


Figure 29: Hypothetical PalmGrid design based on full-scale machine learning. Part of image obtained from He & Sun [29] Deep Residual Learning for Image Recognition. Computer Vision and Pattern Recognition (2015)

3.4 Concluding Remarks

Use of artificial intelligence algorithms revolutionizes the way neuroscientific behavior experiments to be conducted. In current design – despite all deliberately instituted hardware and skillset constraints – PalmGrid is built as an excellent pre-selection method of wall-rearing activities in cylinder test. Its outcomes offer investigators with shrink-wrapped wall-rearing videos to assist their focused assessments of specifics within individual instances. In sum, PalmGrid machine does not replace human investigation. Rather it minimizes investigators expending their precious time resources from reviewing irrelevant video frames, to steer their focus inspecting shrink-wrapped wall-rearing fragments.

In response to the research questions we set out in Section 1.7, we learned that artificial intelligence algorithm anticipates less-than-ideal experimental ambiance and resources constraints, in that a single Pi-camera can fit the purpose in Cylinder Test experiment. This is a big plus for resource hungry laboratories whose purse strings are tightened and skillsets are scarce.

We also learned of the recognition competence in prevalent artificial intelligent algorithms in spite of these hardware and skillset constraints to discern of mice forelimbs. For freely moving mice that maneuver around in the cylinder, recognition in dynamic environment proves the robustness of ResNet algorithms to be used in similar experimental settings.

Independent forelimb recognition has been made possible thanks to the capabilities of DeepLabCut. Even without elaborated hardware calibrations, the software herein designed discerned forelimbs activities independently to extract the respective wall-rearing episodes. Compiling wall-rearing episodic report of respective forelimbs into smaller episodic video

fragments enables focused review of wall-rearing activities. Investigators can now save time to focus on higher quality assessments of these rearing.

Yet the accuracy in posture estimation has rooms for future improvements. Being recognized of forelimb independently is a big leap forward, but higher accuracies of tracking pose another level of challenge. In ResNet-50, we learned that it has 76% accuracy to spot on tracking of digits/palms. Given the non-linear nature of errors introduced into the system in recognition and localization, together with the various constraints that demand the PalmGrid setting to be shrink-wrapped and simple-to-deploy, posture estimation in modest accuracies are expected. To gauge for more accurate wall-rearing episodes, basic engineering principles was used instead of relying on another cascade of artificial intelligence algorithms to yield better outcome accuracies. Using moving averaging techniques, digits coherence, and digits-palm congruence together with statistical outlier removal filters effectively lifted up the accuracies to more acceptable 70%.

Our simplified error propagation model shows that cascading artificial intelligence with the signal processing filters herein designed translates outcome accuracies to roughly 77.78% in gauging wall-rearing episodes. Use of more advanced artificial intelligence and posture extraction algorithms, contingent on lowering graphics processing unit costs, will facilitate higher accuracies in future PalmGrid versions. Likewise, improvements in hardware calibration using anti-reflection acrylic paints or enhancing light ambience will also help to bridge the gap towards 77.78% detection ideals in current configuration.

With wall-rearing episodes compiled into smaller video fragments, different investigators can then leverage the information in wall-rearing episodic reports to compute their respective metric scores in Cylinder Test variants in much reduced processing time. Best of all,

investigators can also perform data mining into specific episodes for further analysis, such that their activities of interests – such as paw-dragging – can be further extracted from these episodes in a fraction of time. Prolonged round-the-clock observations for a large number of subjects are made possible as laborious frame reviews are now machine-enabled, saving precious labor time to more important analytical tasks.

We are certainly embarking on a promising journey, where artificial intelligence and signal processing relieves laborious and time-consuming reviews in Cylinder Test methods. Leveraging our experience learnt in this thesis, we can readily apply the skillsets to benefit other neuroscientific experimentations in likewise manner.

Bibliography

- 1 Lee, B.; Hong, I.; Uhm, Y.; Park, S. The multi-touch system with high applicability using tri-axial coordinate infrared LEDs. *IEEE Trans. Consum. Electron.* 2009, 55, 2416–2424;
- 2 Lim, S.-C.; Shin, J.; Kim, S.-C.; Park, J. Expansion of smart watch touch interface from touchscreen to around device interface using infrared line image sensors. *Sensors* 2015, 15, 16642–16653; [PubMed]
- 3 Han, J.H.; Lee, K.-H.; Han, W.H. A Conclusive Role of Ordinary Transmission for an effective FTIR Touch Screen. In *Proceedings of the IEEE International Conference on Industrial Technology*, Busan, Korea, 26 February–1 March 2014; pp. 583–588;
- 4 Kim, Y.; Park, S.; Park, S.K.; Yun, S.; Kyung, K.-U.; Sun, K. Transparent and flexible force sensor array based on optical waveguide. *Opt. Express* 2012, 20, 14486–14493; [PubMed]
- 5 Shikida, M.; Asano, K. A flexible transparent touch panel based on ionic liquid channel. *IEEE Sens. J.* 2013, 13, 3490–3495;
- 6 Kamali, B. Touch-screen Displays. In *Instrument Engineers' Handbook—Process Control and Optimization*, 4th ed.; Lipták, B.G., Ed.; CRC Press: Boca Raton, FL, USA, 2006; pp. 845–853;
- 7 Nakamura, T.; Yamamoto, A. Interaction force estimation on a built-in position sensor for an electrostatic visual-haptic display. *ROBOMECH J.* 2016, 3, 1–11;
- 8 Kim, W.; Oh, H.; Kwak, Y.; Park, K.; Ju, B.-K.; Kim, K. Development of a carbon nanotube-based touchscreen capable of multi-touch and multi-force sensing. *Sensors* 2015, 15, 28732–28741; [PubMed]
- 9 Wang, B.; Long, J.; Teo, K.H. Multi-channel capacitive sensor arrays. *Sensors* 2016, 16, 150; [PubMed]
- 10 Chen, Y.-L.; Liang, W.-Y.; Chiang, C.-Y.; Hsieh, T.-J.; Lee, D.-C.; Yuan, S.-M.; Chang, Y.-L. Vision-based finger detection, tracking, and event identification techniques for multi-touch sensing and display systems. *Sensors* 2011, 11, 6868–6892; [PubMed]
- 11 Reis, S.; Correia, V.; Martins, M.; Barbosa, G.; Sousa, R.M.; Minas, G.; Lanceros-Mendez, S.; Rocha, J.G. Touchscreen Based on Acoustic Pulse Recognition with Piezoelectric Polymer Sensors. In *Proceedings of the IEEE International Symposium on Industrial Electronics*, Bari, Italy, 4–7 July 2010;
- 12 Katsuki, T.; Nakazawa, F.; Sano, S.; Takahashi, Y.; Satoh, Y. A compact and High Optical Transmission SAW Touch Screen with ZnO Thin-Film Piezoelectric Transducers. In *Proceedings of the 2003 IEEE Symposium on Ultrasonics*, Honolulu, HI, USA, 5–8 October 2003; pp. 821–824;
- 13 Liu, Y.; Nikolovski, J.P.; Mechbal, N.; Hafez, M.; Vergé, M. An acoustic multi-touch sensing method using amplitude disturbed ultrasonic wave diffraction patterns. *Sens. Actuators A Phys.* 2010, 162, 394–399;
- 14 Kurita, K.; Fujii, Y.; Shimada, K. A new technique for touch sensing based on measurement of current generated by electrostatic induction. *Sens. Actuators A Phys.* 2011, 170, 66–71;
- 15 Shinoda, H.; Chigusa, H.; Makino, Y. Flexible tactile sensor skin using wireless sensor elements coupled with 2D microwaves. *J. Robot. Mechatron.* 2010, 22, 784–789;
- 16 Dahiya, R.S.; Adami, A.; Collini, C.; Lorenzelli, L. POSFET tactile sensing arrays using CMOS technology. *Sens. Actuators A Phys.* 2012, 47, 894–897;
- 17 Walker, G. A review of technologies for sensing contact location on the surface of a display. *J. Soc. Inf. Disp.* 2012, 20, 413–440;
- 18 Tsuji, S.; Kohama, T. A layered 3D touch screen using capacitance measurement. *IEEE Sens. J.* 2014, 14, 3040–3045;
- 19 Dirk J. Ardesch; Matilde Balbi; Timothy H. Murphy. Automated touch sensing in the mouse tapered beam test using Raspberry Pi. *Journal of Neuroscience Methods* 291 (2017) 221–226;
- 20 Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation* 2006 Vol. 18, 1527-1554;
- 21 Canziani A., Culurciello E., Paszke A. An analysis of Deep Neural Networks for practical applications. *Computer Vision and Pattern Recognition* April 2017. arXiv:1605.07678;
- 22 Sabour S, Frosst N., Hinton G. Dynamic Routing Between Capsules. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA. arXiv:1710.09829v2 [cs.CV] 7 Nov 2017;
- 23 Pishchulin L., Insafutdinov E., Tang S., Andres B., Andriluka M., Gehler P., and Schiele B. DeepCut: Joint Subset Partition and Labeling for Multi-Person Pose Estimation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*. arXiv:1511.06645 [cs.CV] Apr 2016.
- 24 Mathis A., Mamidanna P., Cury KM, Abe T., Murthy VN, Mathis MW, Matthias Bethgea. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience* volume 21, pages1281–1289 (2018). <https://www.nature.com/articles/s41593-018-0209-y>;

- 25 Schallert T., Fleming S., Leasure JL, Tillerson JL, Sondra T., Bland. CNS plasticity and assessment of forelimb sensorimotor outcome in unilateral rat models of stroke, cortical ablation, parkinsonism, and spinal cord injury. *Neuropharmacology* Volume 39, Issue 5, April 2000, Pages 777-787. [https://doi.org/10.1016/S0028-3908\(00\)00005-8](https://doi.org/10.1016/S0028-3908(00)00005-8);
- 26 Xiaoling Li, Kathleen K. Blizzard, Zhiyuan Zeng, A. Courtney DeVries, Patricia D. Hurn, Louise D. McCullough. Chronic behavioral testing after focal ischemia in the mouse: functional recovery and the effects of gender. *Experimental Neurology* Volume 187, Issue 1, May 2004, Pages 94-104. <https://doi.org/10.1016/j.expneurol.2004.01.004>;
- 27 Shanina EV, Schallert T, Witte OW, Redecker C. Behavioral recovery from unilateral photothrombotic infarcts of the forelimb sensorimotor cortex in rats: Role of the contralateral cortex. *Neuroscience* Volume 139, Issue 4, 2006, Pages 1495-1506. <https://doi.org/10.1016/j.neuroscience.2006.01.016>;
- 28 Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015
- 29 He, Zhang, Ren, Sun. Deep Residual Learning for Image Recognition. *Computer Vision and Pattern Recognition*. Dec. 2015, arXiv:1512.03385
- 30 Roome R., Vanderluit J. Paw-dragging: a novel, sensitive analysis of the mouse cylinder test. *J. Vis. Exp.* (98), e52701. Doi:10.3791/52701 (2015)
- 31 Insafutdinov, Pishchulin, Andres, Andriluka, and Schiele. DeeperCut: A Deeper, Stronger, and Faster Multi-person Pose Estimation Model (2016) *ECCV 2016, Part VI, LNCS 9910*, pp. 34–50, 2016. DOI: 10.1007/978-3-319-46466-4_3
- 32 Bulman-Fleming, Bryden, Rogers. Mouse paw preference: effects of variations in testing protocol. *Oct 1996, Behavioural Brain Research* 86 (1997) 79–87.
- 33 Cunha, Esteves, das Neves, Borges, Guimarães, Sousa, Almeida and Leite-Almeida. Pawedness Trait Test (PaTRaT)—A New Paradigm to Evaluate Paw Preference and Dexterity in Rats. *Frontiers in Behavioral Neuroscience*. doi: 10.3389/fnbeh.2017.00192.
- 34 Collins, R. L. (1968). On the inheritance of handedness. I. Laterality in inbred mice. *J. Hered.* 59, 9–12. doi: 10.1093/oxfordjournals.jhered.a107656

Appendices

Appendix A – PalmGrid Hardware and Settings

A.1 Hardware Components

Components

Raspberry Pi

Pictures



Pi with Video Cam



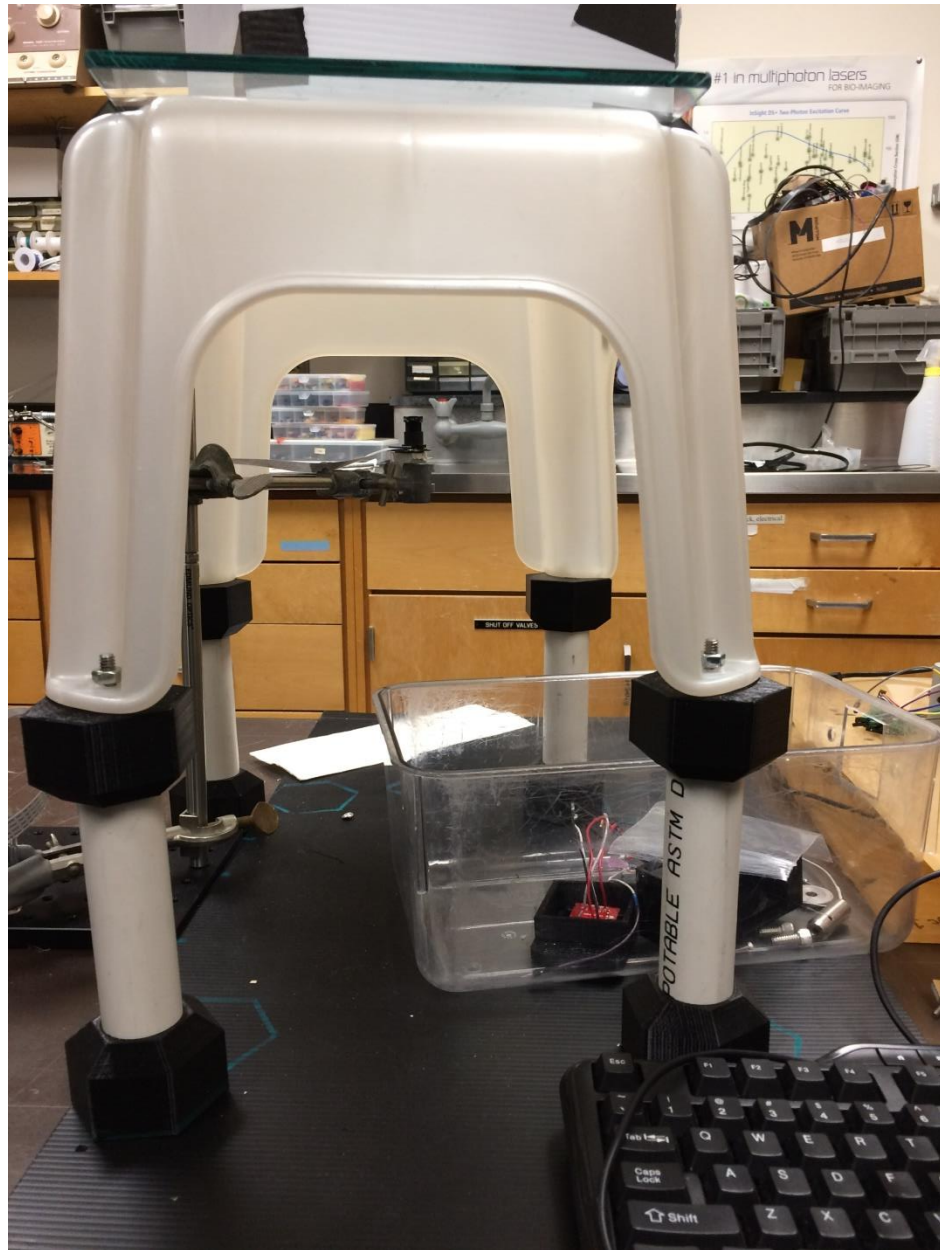
Cylinder for
Cylinder Test



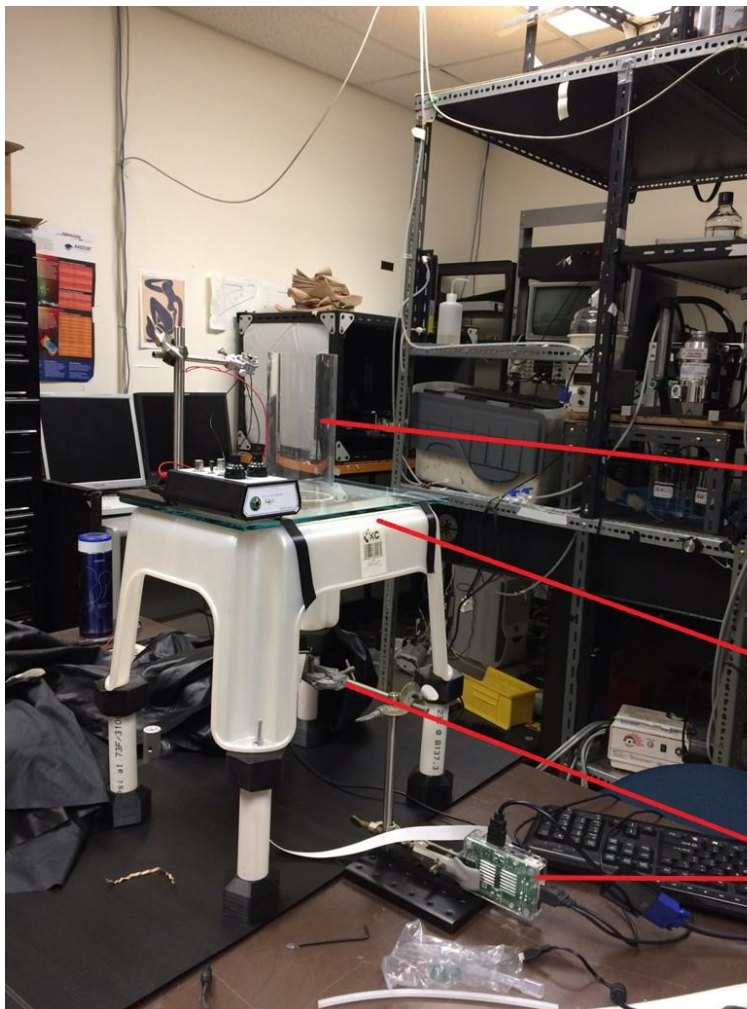
Components

Pictures

Transparent Stool



A.2 PalmGrid Experimentation Setup



Cylinder

Transparent Stool

Raspberry Pi & basic
camera system

A.2 Raspberry Pi Video Camera version 2 Specification

	Camera Module v2
Weight	3g
Still resolution	8 Megapixels
Video modes	1080p30, 720p60 and 640 × 480p60/90
Linux integration	V4L2 driver available
C programming API	OpenMAX IL and others available
Sensor	Sony IMX219
Sensor resolution	3280 × 2464 pixels
Sensor image area	3.68 x 2.76 mm (4.6 mm diagonal)
Pixel size	1.12 μm x 1.12 μm
Optical size	1/4"
Focal length	3.04 mm
Horizontal field of view	62.2 degrees
Vertical field of view	48.8 degrees
Focal ratio (F-Stop)	2

A.3 Raspberry Pi Video Recording Scripts

```
raspivid -t 210000 -md 6 -fps 25 -o <pivideo>.h264
```

The command above requests raspberry pi to record video for 3.5 minutes in 1200 x 1200 pixels in 16:9 aspect ratio; with pixels at 25 frames per second and output the file to <pivideo.h264>.

Appendix B – PalmGrid Signal Processing and Gauging Module Pseudocodes

B.1 Coherent Fragment Extraction

Procedure Coherent

For $i := 1$ to Number of Slow Moving Fragments

 Identify at least two slowest moving fingertips of the given forelimb;

 For those fragments identified

 Gauge for Sub-fragments that movements are accounted for as slow-moving;

 If two sub-fragments separate among themselves by 1 frame, merge the two;

 For each sub-fragment

 Compute Statistics for each, in changes of radial displacements $\pm 0.5s$ before and after each sanitized sub-fragment;

 Retain those sub-fragment(s) that approach the cylinder wall before the sub-fragment, and retracing/retained from wall thereafter;

 End

 End

End

B.2 Congruence Points Identification

Procedure Congruence

For $i := 1$ to Number of Slow Moving Fragments

 Identify three slowest moving fingertips of the given forelimb;

 For the three identified, slow-moving fingertips

 Gauge for Sub-fragments whose fingertips to palm distance came to minima;

 End

End

B.3 Refined Sub-Fragment Gauging

Procedure Decision

Merge coherent sub-fragments with congruence points;

For each sub-fragment of both coherence and congruence

 Compute statistics for each;

 Filter out non-wall-rearing based on meeting one of the following criteria:

1. If the wall-rearing durations is less than 0.1 seconds, they are regarded as transient touches and not counted as a separate wall-rearing episode;
2. Adjacent fingertips are not distant from each other for more than 100 pixels in any one time;
3. Any plausible wall-rearing episode whose standard deviation of the entire fragment is less than 10 pixels (2% about mean) is taken as stationary.

End

Consolidate retained sub-fragments snapshots that are nearby each other within 2 seconds into Sub-fragment episodes;

Appendix C – Test Results

C.1 Overall Left and Right Forelimbs taken together

		Actual	Congruence Filter Performance			
			Inside CZ			Outside CZ
			Correctly	False +ve	False -ve	
EMX01-01	Left	27	18	9	4	
	Right	23	14	9	4	
EMX01-02	Left	15	11	4	1	
	Right	11	5	6	8	
EMX02-01	Left	16	13	3	1	
	Right	18	8	10	5	
EMX02-03	Left	16	8	8	2	
	Right	11	4	7	5	
EMX03-02	Left	52	41	11	1	
	Right	53	40	13	3	
EMX03-03	Left	45	36	9	1	
	Right	43	35	8	5	
EMX04-02	Left	32	23	9		
	Right	29	19	10	5	
EMX04-03	Left	42	32	10	2	
	Right	51	34	17	1	
EMX06-01	Left	45	33	12		
	Right	26	20	6	14	
EMX06-02	Left	9	6	3	2	
	Right	16	6	10	4	
Total		580	406	174	68	0
Success Rate%			70.0%			
False Positives%				90%		
Undetected %					12%	

Correct	Algorithmic rules detected there should be at least one Touch within the slow moving fragment; and actually there are
False +ve	Algorithmic rules detected that there should be at least one Touch within slow moving fragment; but actually No Touch
False -ve	Algorithm did not detect anything within the slow moving fragment; but actually there was at least one Touch
Outside CZ	Actual Touch that lies beyond Congruence Zone Detections; usually due to obstruction, start of video frame etc.
Success Rate %	Total Number of Actual Wall Rearing / Number of detected Hits
False Positive %	Total Number of False Positives / Number of detected Hits
Undetected %	Total Number of Undetected Hits / Total Number of detected Hits

C.2 Overall Left and Right Forelimbs taken together in Percentage

		Congruence Filter Performance				
		Actual	Inside CZ			Outside CZ
		Touches	Correctly	False +ve	False -ve	
EMX01-01	Left	27	67%	33%	15%	
	Right	23	61%	39%	17%	
EMX01-02	Left	15	73%	27%	7%	
	Right	11	45%	55%	73%	
EMX02-01	Left	16	81%	19%	6%	
	Right	18	44%	56%	28%	
EMX02-03	Left	16	50%	50%	13%	
	Right	11	36%	64%	45%	
EMX03-02	Left	52	79%	21%	2%	
	Right	53	75%	25%	6%	
EMX03-03	Left	45	80%	20%	2%	
	Right	43	81%	19%	12%	
EMX04-02	Left	32	72%	28%		
	Right	29	66%	34%	17%	
EMX04-03	Left	42	76%	24%	5%	
	Right	51	67%	33%	2%	
EMX06-01	Left	45	73%	27%		
	Right	26	77%	23%	54%	
EMX06-02	Left	9	67%	33%	22%	
	Right	16	38%	63%	25%	
Total		580	406	174	68	0
Success Rate%			70.0%			
False Positives%				30%		
Undetected %					12%	

Correct	Algorithmic rules detected there should be at least one Touch within the slow moving fragment; and actually there are
False +ve	Algorithmic rules detected that there should be at least one Touch within slow moving fragment; but actually No Touch
False -ve	Algorithm did not detect anything within the slow moving fragment; but actually there was at least one Touch
Outside CZ	Actual Touch that lies beyond Congruence Zone Detections; usually due to obstruction, start of video frame etc.
Success Rate %	Total Number of Actual Wall Rearing / Number of detected Hits
False Positive %	Total Number of False Positives / Number of detected Hits
Undetected %	Total Number of Undetected Hits / Total Number of detected Hits

C.3 Right Forelimb Only in Number of Touch

		Actual	Congruence Filter Performance			
			Inside CZ			Outside CZ
			Correctly	False +ve	False -ve	
EMX01-01	Right	23	14	9	4	
EMX01-02	Right	11	5	6	8	
EMX02-01	Right	18	8	10	5	
EMX02-03	Right	11	4	7	5	
EMX03-02	Right	53	40	13	3	
EMX03-03	Right	43	35	8	5	
EMX04-01	Right	29	19	10	5	
EMX04-03	Right	51	34	17	1	
EMX06-01	Right	26	20	6	14	
EMX06-02	Right	16	6	10	4	
Total		281	183	96	54	0
Success Rate%			65.8%			
False Positives%				34%		
Undetected %						19%

Correct		Algorithmic rules detected there should be at least one Touch within the slow moving fragment; and actually there are
False +ve		Algorithmic rules detected that there should be at least one Touch within slow moving fragment; but actually No Touch
False -ve		Algorithm did not detect anything within the slow moving fragment; but actually there was at least one Touch
Outside CZ		Actual Touch that lies beyond Congruence Zone Detections; usually due to obstruction, start of video frame etc.
Success Rate %	Total Number of Actual Wall Rearing / Number of detected Hits	
False Positive %	Total Number of False Positives / Number of detected Hits	
Undetected %	Total Number of Undetected Hits / Total Number of detected Hits	

C.4 Right Forelimb Only in % of Touch

		Actual	Congruence Filter Performance			
			Inside CZ			Outside CZ
			Correctly	False +ve	False -ve	
EMX01-01	Right	23	61%	39%	17%	
EMX01-02	Right	11	45%	55%	73%	
EMX02-01	Right	16	50%	63%	31%	
EMX02-03	Right	11	36%	64%	45%	
EMX03-02	Right	53	75%	25%	6%	
EMX03-03	Right	43	81%	19%	12%	
EMX04-01	Right	29	66%	34%	17%	
EMX04-03	Right	51	67%	33%	2%	
EMX06-01	Right	26	77%	23%	54%	
EMX06-02	Right	16	38%	63%	25%	
Total		279	190	89	50	
Success Rate%			68.1%			
False Positives%				32%		
Undetected %						18%

Correct		Algorithmic rules detected there should be at least one Touch within the slow moving fragment; and actually there are
False +ve		Algorithmic rules detected that there should be at least one Touch within slow moving fragment; but actually No Touch
False -ve		Algorithm did not detect anything within the slow moving fragment; but actually there was at least one Touch
Outside CZ		Actual Touch that lies beyond Congruence Zone Detections; usually due to obstruction, start of video frame etc.
Success Rate %	Total Number of Actual Wall Rearing / Number of detected Hits	
False Positive %	Total Number of False Positives / Number of detected Hits	
Undetected %	Total Number of Undetected Hits / Total Number of detected Hits	

C.5 Left Forelimb Only in Number of Touch

		Actual	Congruence Filter Performance			
			Inside CZ			Outside CZ
			Correctly	False +ve	False -ve	
EMX01-01	Left	27	18	9	4	
EMX01-02	Left	15	11	4	1	
EMX02-01	Left	16	13	3	1	
EMX02-03	Left	16	8	8	2	
EMX03-02	Left	52	41	11	1	
EMX03-03	Left	45	36	9	1	
EMX04-01	Left	32	23	9	0	
EMX04-03	Left	42	32	10	2	
EMX06-01	Left	45	33	12	0	
EMX06-02	Left	9	6	3	2	
Total		299	221	78	14	0
Success Rate%			73.9%			
False Positives%				26%		
Undetected %					5%	

Correct		Algorithmic rules detected there should be at least one Touch within the slow moving fragment; and actually there are
False +ve		Algorithmic rules detected that there should be at least one Touch within slow moving fragment; but actually No Touch
False -ve		Algorithm did not detect anything within the slow moving fragment; but actually there was at least one Touch
Outside CZ		Actual Touch that lies beyond Congruence Zone Detections; usually due to obstruction, start of video frame etc.
Success Rate %	Total Number of Actual Wall Rearing / Number of detected Hits	
False Positive %	Total Number of False Positives / Number of detected Hits	
Undetected %	Total Number of Undetected Hits / Total Number of detected Hits	

C.6 Left Forelimb Only in % of Touch

		Actual	Congruence Filter Performance			
			Inside CZ			Outside CZ
			Correctly	False +ve	False -ve	
EMX01-01	Left	27	67%	33%	13%	
EMX01-02	Left	15	73%	27%	7%	
EMX02-01	Left	16	81%	19%	6%	
EMX02-03	Left	16	50%	50%	13%	
EMX03-02	Left	32	79%	21%	2%	
EMX03-03	Left	45	80%	20%	2%	
EMX04-01	Left	32	72%	28%	0%	
EMX04-03	Left	42	76%	24%	3%	
EMX06-01	Left	45	73%	27%	0%	
EMX06-02	Left	9	67%	33%	22%	
Total		299	221	78	14	0
Success Rate%			73.9%			
False Positives%				26%		
Undetected %					5%	

Correct		Algorithmic rules detected there should be at least one Touch within the slow moving fragment; and actually there are
False +ve		Algorithmic rules detected that there should be at least one Touch within slow moving fragment; but actually No Touch
False -ve		Algorithm did not detect anything within the slow moving fragment; but actually there was at least one Touch
Outside CZ		Actual Touch that lies beyond Congruence Zone Detections; usually due to obstruction, start of video frame etc.
Success Rate %		Total Number of Actual Wall Rearing / Number of detected Hits
False Positive %		Total Number of False Positives / Number of detected Hits
Undetected %		Total Number of Undetected Hits / Total Number of detected Hits

Appendix D: Methods for Forelimb Tests

D.1 Cylinder Test (Li & McCullough (2004))

The cylinder test was adapted for use in mouse to assess forelimb use and rotation asymmetry. The mouse was placed in a transparent cylinder 9-cm diameter and 15 cm in height and videotaped during the test. A mirror was placed behind the cylinder with an angle to enable the rater to record forelimb movements when the mouse was turned away from the camera. After the mouse was put into the cylinder, forelimb use of the first contact against the wall after rearing and during lateral exploration was recorded by the following criteria:

- (1) The first forelimb to contact the wall during a full rear was recorded as an independent wall placement for that limb.
- (2) Simultaneous use of both the left and right forelimb by contacting the wall of the cylinder during a full rear and for lateral movements along the wall was recorded as “both” movement.
- (3) After the first forelimb (for example right forelimb) contacted the wall and then the other forelimb was placed on the wall, but the right forelimb was not removed from the wall, a “right forelimb independent” movement and a “both” movement were recorded. However, if the other (left forelimb) made several contacting movements on the wall, a “right forelimb independent” movement and only one “both” movement was recorded.
- (4) When the mouse explored the wall laterally, alternating both forelimbs, it was recorded as a “both” movement. A total of 20 movements were recorded during the 10-min test to compute a final score by the following formula:

$$\text{Final Score} = \frac{(\text{nonimpaired forelimb movement} - \text{impaired forelimb movement})}{(\text{nonimpaired forelimb movement} + \text{impaired forelimb movement} + \text{both forelimb movements})}$$

Non-impaired forelimb movement	=	Independent left / right forelimb wall-rearing before stroke
Impaired forelimb movement	=	Independent left / right wall-rearing count post-stroke
Both movements	=	1) Simultaneous left/right wall rearing 2) left/right alternate exploration

Figure 30: Cylinder Test Score Calculation

This test evaluates forelimb use asymmetry for weight shifting during vertical exploration and provides high interrater reliability even with inexperienced raters. Occasionally mice with large deficits did not move frequently enough to obtain an adequate number of vertical movements, these animals recovered later in testing and to avoid bias these animals were unscored until they could perform the test.

D.2 Forelimb preference and sliding test (Shanina & Redecca 2006)

As a variant of cylinder test, forelimb use during spontaneous vertical exploration was analyzed based on the method described by Schallert et al. (2000).

The rats were videotaped in a transparent glass cylinder for 5–7 min depending on the degree of activity during the trial (Fig. 1B). Two mirrors combined at an angle of 90° were placed behind the glass cylinder allowing the recording of forelimb movements even when the animal turned away from the camera. Several behavioral elements were scored to determine the extent of forelimb impairment during spontaneous exploration of the glass cylinder. The independent or simultaneous use of the left or right forelimb was analyzed

- a) at first contact with the wall; and
- b) during vertical and horizontal movements along the wall; and
- c) sliding movements of each forelimb at the wall of the cylinder were scored

Forelimb activity (FLA) was evaluated for each forelimb using the following formula:

$$FLA = \frac{\text{first contact} + \text{horizontal} + \text{vertical}}{\text{number of rearings}}$$

Figure 31: Forelimb Activity Score Formula

In addition, the frequency of sliding movements (%) which occurred during vertical activity at the wall of the glass cylinder was assessed for each forelimb using the following score:

$$\text{sliding score (\%)} = \frac{\text{sliding}}{\text{first contact} + \text{horizontal} + \text{vertical}} \times 100$$

In order to illustrate the time course of alterations in these behavioral tests data are in part given as percentage difference between preoperative baseline and different time points after the infarcts.

D.3 Cylinder Test (Schallert (2000))

Forelimb use during explorative activity was analyzed by videotaping rats in a transparent cylinder (20 cm diameter and 30 cm height) for 3–10 min depending on the degree of movement maintained during the trial. A mirror was placed behind the cylinder at an angle to enable the rater to record forelimb movements when the animal was turned away from the camera. The cylindrical shape encourages vertical exploration of the walls with the forelimbs as well as landing activity. The cylinder was high enough that the animal could not reach the top edge by rearing and wide enough to permit a 2 cm space between the tip of the snout and the base of the tail when the animal was not rearing. However, other chambers such as the home cage may be used as long as the behavior of the animal can be viewed unobstructed from all directions. All scoring was done by an experimenter blind to the condition of the animal using a VCR with slow motion and frame by frame capabilities. An advantage of the limb use asymmetry (cylinder) test is that inter-rater reliability is very high ($r > 0.95$) even with relatively inexperienced raters. Following some types of injury the animals may not move frequently enough to obtain an adequate number of vertical movements. In this case videotaping in the home cage at the beginning of the dark cycle may be necessary.

Several behaviors were scored to determine the extent of forelimb-use asymmetry displayed by the animal. These behaviors were recorded during vertical movements along the wall and for landings after a rear:

(a) independent use of the left or right forelimb for contacting the wall during a full rear, to initiate a weightshifting movement or to regain center of gravity while moving laterally in a vertical posture;

(b) independent use of the left or right forelimb to land after a rear:

(c) simultaneous use of both the left and right forelimb for contacting the wall of the cylinder during a full rear and for lateral movements along the wall;

(d) simultaneous use of both the left and right forelimb for landing following a rear. If a rater could not determine whether one limb was being used independently or simultaneously, that movement was not scored. Each behavior was expressed in terms of

(a) percent use of the non-impaired forelimb relative to the total number of limb use observations (impaired, unimpaired and both limb use observations) for wall movements;

(b) percent use of the impaired forelimb relative to the total number of limb use observations for wall movements;

(c) percent use of the limbs simultaneously relative to the total number of limb use observations for wall movements;

(d) percent use of the non-impaired forelimb relative to the total number of limb use observations for landings:

(e) percent use of the impaired forelimb relative to the total number of limb use observations for landings; and

(f) percent simultaneous limb use observations relative to the total number of limb use observations for landings.

Wall-associated ratios and landing ratios can be averaged together for scores that reflect equal contributions from asymmetries in wall movements and landings.

D.4 Paw-Dragging Method

Paw dragging method is essentially the same as the above cylinder tests, except it focuses on quantifying the number of paw-drags from the recorded videos of mice. Paw-dragging behaviour is distinct from normal paw touches as follows:

1. If the paw contacts the cylinder wall with a full open palm, it will slowly fall away from the wall, often with a slight tremor. The movement begins with the digits dragging against the cylinder wall either in a medial or downward direction, before falling away completely. The mouse will then dismount with its unaffected paw before landing on all fours. This is considered a paw-drag and should be counted in a tally.
2. If the paw does not contact the cylinder wall with a fully open palm, it will graze the cylinder wall with its digits before falling away from the cylinder wall. Similarly, a mouse may drag its paw against the cylinder wall but not release it entirely before dismounting. These are both considered paw-drags as well as touches and should be counted as both in a tally.
3. The paw may also drag along the cylinder wall while a mouse explores the cylinder. In this case, the paw will follow the twisting of the mouse's torso as it explores left or right of its original position before dismounting. This is not considered a paw-drag, as it depends on the mouse randomly choosing a direction to explore and does not depend on which cortical hemisphere was damaged.

Paw-drags are expressed as a percentage of paw-drags per total number of paw touches during a session. Express the number of paw-drags as a percentage of total paw contacts for each forelimb separately.

Appendix E: PalmGrid Station Installation Manual

E.1 Python 3.6

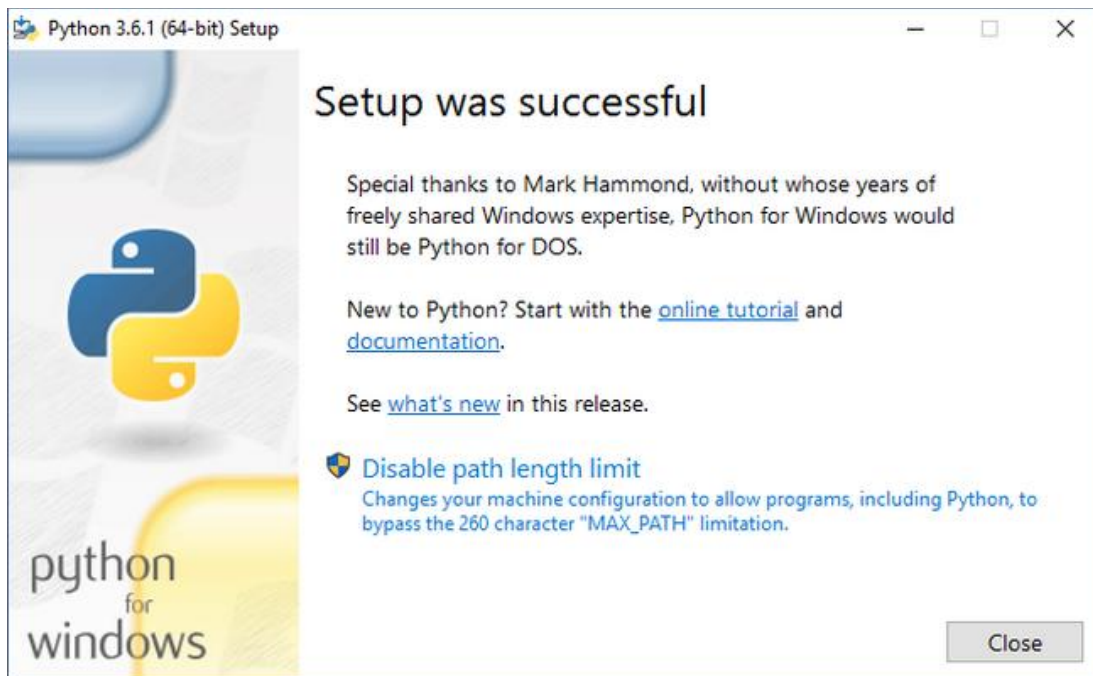
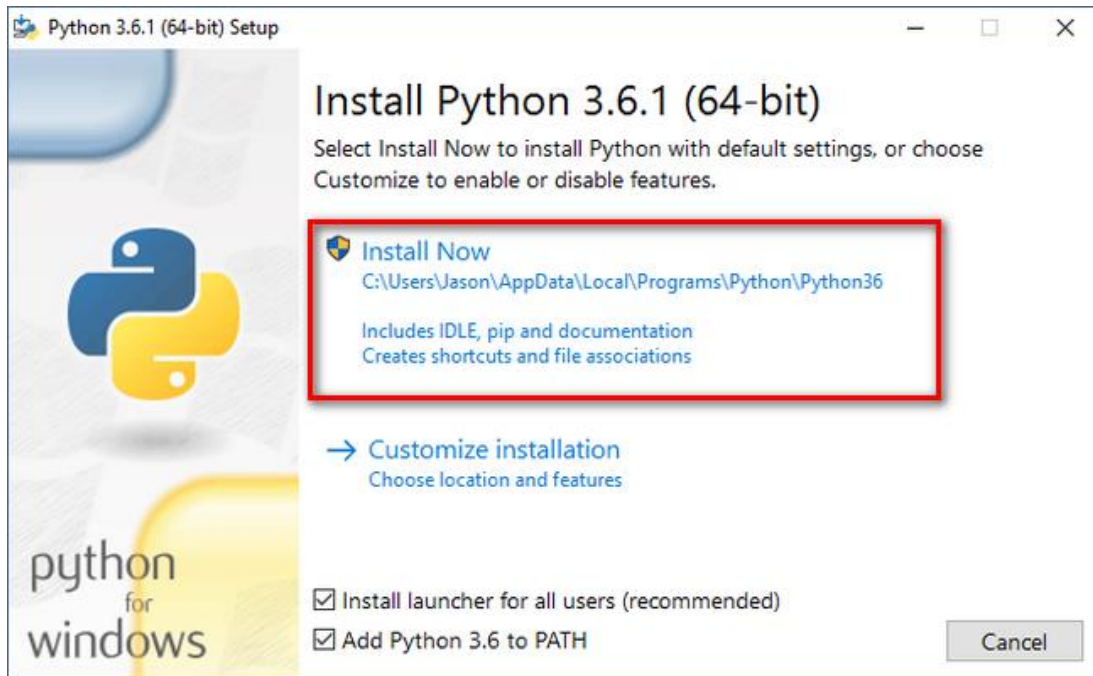
1. Download python 3.6

Under the main entry for both versions you'll see an "x86-64" installer, as seen below.

- [Python 3.6.1 - 2017-03-21](#)
 - Download [Windows x86 web-based installer](#)
 - Download [Windows x86 executable installer](#)
 - Download [Windows x86 embeddable zip file](#)
 - Download [Windows x86-64 web-based installer](#)
 - Download [Windows x86-64 executable installer](#)
 - Download [Windows x86-64 embeddable zip file](#)
 - Download [Windows help file](#)

2. Extract the software package, and hit setup

3. Enable the "Add Python 3.6 to PATH" option and then click "Install Now."



4. Check for correct installation using `python -v` that we used above to check that it is installed correctly and the path variable is set.

E.2 DeepLabCut

There are several modes of installation, and the user should decide to either use a **system-wide, Anaconda environment** based installation (recommended). One can of course also use other Python distributions than Anaconda, but this is the easiest route.

All the following commands will be run in the `cmd` in Windows. Please first open the terminal (search `cmd`).

• **Anaconda:**

Anaconda is perhaps the easiest way to install Python and additional packages across various operating systems. First create an Anaconda environment. With Anaconda you create all the dependencies in an environment on your machine in the following way. More details can be found in the conda environment readme.

Windows:

DeepLabCut provides environment files for Windows. They can be installed by typing (from the terminal, within in this conda-environments folder): `conda env create -f dlc-windowsCPU.yaml` or `conda env create -f dlc-windowsGPU.yaml` for the GPU version. See further details in this issue.

Then,

Windows: `pip install -U wxPython`

Install TensorFlow with GPU support:

1. Install [TensorFlow](#). In the Nature Neuroscience paper TensorFlow 1.0 with CUDA (Cuda 8.0) was used. Some other versions of TensorFlow have been tested, but they are not tested! (i.e. these versions have been tested 1.2, 1.4, 1.8 or 1.10-1.13, but might require different CUDA versions)! Please check your driver/cuDNN/CUDA/TensorFlow versions [on this Stackoverflow post](#).
2. **Install the NVIDIA CUDA package and an appropriate driver for your specific GPU.** Please follow the instructions found here: <https://www.tensorflow.org/install/gpu>, and more [tips below](#). The order of operations matters.

3. Some tips for installing **TensorFlow 1.8** will follow here:

FIRST, install a driver for your GPU (we recommend the 384.xx) Find DRIVER HERE: <https://www.nvidia.com/download/index.aspx>

- check which driver is installed by typing this into the terminal: `nvidia-smi`

SECOND, install CUDA (9.0 here): <https://developer.nvidia.com/cuda-90-download-archive>

THIRD, install TensorFlow:

Package for pip install:

```
pip install tensorflow-gpu==1.8 —with GPU support (Ubuntu and Windows)
```

Note, the version is specified by using: `==1.8`

FOURTH, Please check your CUDA and [TensorFlow installation](#) with the lines below:

Start a python session: `ipython`

```
import tensorflow as tf
```

```
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
```

You can test that your GPU is being properly engaged with these additional [tips](#).

Troubleshooting:

TensorFlow: Here are some additional resources users have found helpful (posted without endorsement):

- <https://stackoverflow.com/questions/30820513/what-is-the-correct-version-of-cuda-for-my-nvidia-driver/30820690>
- <https://stackoverflow.com/questions/50622525/which-tensorflow-and-cuda-version-combinations-are-compatible>
- <http://blog.nitishmutha.com/tensorflow/2017/01/22/TensorFlow-with-gpu-for-windows.html>
- <https://developer.nvidia.com/cuda-toolkit-archive>

- <http://www.python36.com/install-tensorflow-gpu-windows/>

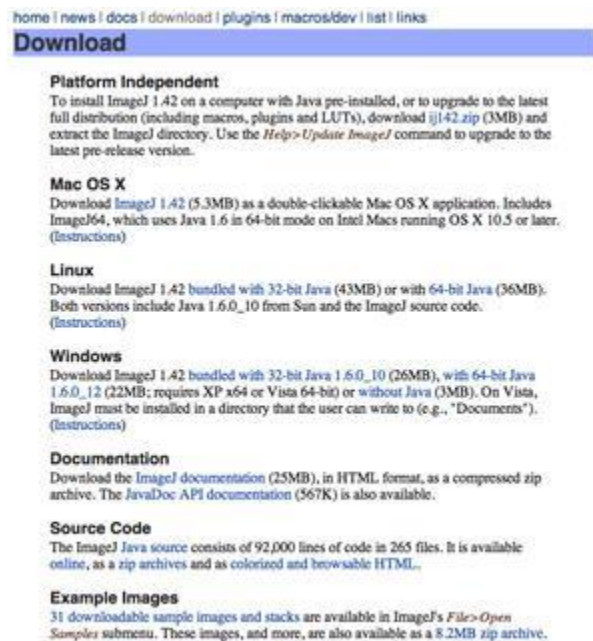
• **System-wide considerations:**

If you perform the system wide installation, and the computer has other Python packages or TensorFlow versions installed that conflict, this will overwrite them. If you have a dedicated machine for DeepLabCut, this is fine. If there are other applications that require different versions of libraries, then one would potentially break those applications. The solution to this problem is to create a virtual environment, a self-contained directory that contains a Python installation for a particular version of Python, plus additional packages. One way to manage virtual environments is to use conda environments (for which you need Anaconda installed).

E.3 ImageJ

1) – Download and Install ImageJ Software

ImageJ is in the public domain. It can be freely downloaded and installed on any computer including those at schools, homes, and businesses.

The screenshot shows the 'Download' section of the ImageJ website. At the top, there is a navigation bar with links: 'home | news | docs | download | plugins | macros/dev | list | links'. Below this, the 'Download' heading is highlighted in a blue box. The page is organized into several sections: 'Platform Independent' (describing the full distribution and update process), 'Mac OS X' (providing download links for ImageJ 1.42 and Image64), 'Linux' (providing download links for 32-bit and 64-bit Java), 'Windows' (providing download links for 32-bit and 64-bit Java), 'Documentation' (providing download links for HTML and JavaDoc API documentation), 'Source Code' (providing download links for the Java source code), and 'Example Images' (providing download links for sample images and stacks).

ImageJ download page.

Go to the [ImageJ Download page](#) , and download and install the application for your operating system.

Note to Windows Users: It is recommended that you install ImageJ in the Documents directory, rather than in the Program Files directory. For security reasons, Windows 7 and Windows Vista do not allow programs to alter themselves by writing files to the Program Files directory. If ImageJ is installed in the Program Files directory, then the update function in Step 2 below will not work properly. In addition, if you are a Windows Vista user, be sure to choose the correct version of ImageJ (either 32-bit or 64-bit) for your computer.

E.4 ffmpeg

1. Download a static build from [here](#).
2. Use [7-Zip](#) to unpack it in the folder of your choice.
3. [Open a command prompt with administrator's rights](#).
NOTE: Use CMD.exe, do not use Powershell! The syntax for accessing environment variables is different from the command shown in Step 4 - running it in Powershell will overwrite your System PATH with a bad value.
4. Run the command (see note below; in Win7 and Win10, you might want to use the Environmental Variables area of the Windows Control Panel to update PATH):
`setx /M PATH "path\to\ffmpeg\bin;%PATH%"`

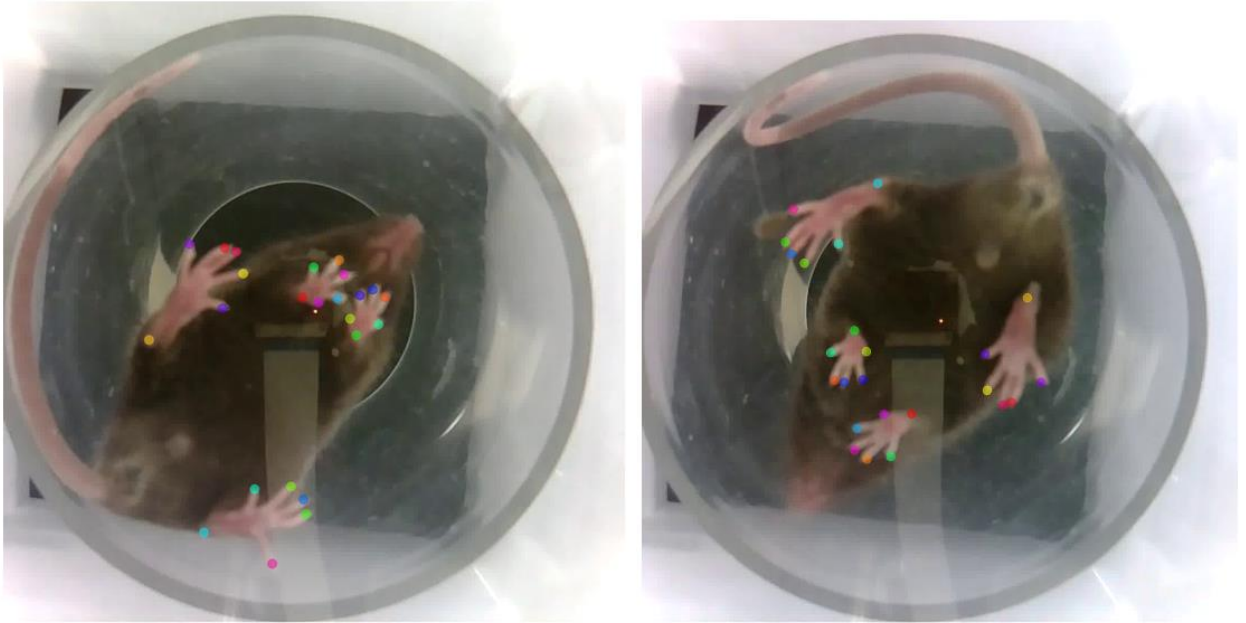
NB: Do not run setx if you have more than 1024 characters in your system PATH variable. See [this post on SuperUser](#) that discusses alternatives. Be sure to alter the command so that path\to reflects the folder path from your root to ffmpeg\bin.

Appendix F: Labeling Forelimbs of Mouse Images that trains

DeepLabCut

To train DeepLabCut for PalmGrid, we recommend labeling maximally diverse images (i.e., different poses) in a consistent, anticlockwise manner and curating the labeled data well. In our experience, we expanded the initial training dataset in an iterative fashion.

1. First, set up the Cylinder Test experiment. Expose the setting with white cardboard encasing the cylinder with black cardboard covering the top of cylinder. These offer maximum light contrasts to the mouse subjects.
2. Second, record the video of 1200 x 1200 pixel resolutions using Raspberry Pi
3. Third, Convert the recorded video using ffmpeg to .wmv format, where DeepLabCut requires to generate a training set.
4. Fourth, select frames where reliably captured behaviors and avoided those corrupted with noise (e.g. blurred images when displayed in ImageJ),
5. Fifth, in our application we label BOTH forelimbs and rear limbs using ImageJ. ***Label 6 points (namely the five digits and the palm) for each limb and curate each selected frames in anticlockwise manner.*** An example of post-labeled frame is illustrated below.



NB. In our motivation example as we do not know whether rear limb coordinates will be utilized. On hindsight, labeling forelimbs will be good enough. We learnt that labeling has to be done in a consistent, anticlockwise manner as recommended by DeepLabCut

6. Sixth, run ImageJ's Analyze / Measure Tools to extract coordinates of these labeled limbs. Save the results of these measurements as directed by DeepLabCut user manual to train DeepLabCut server.
7. Try a few samples of video analysis. If the labeling was bad or inconsistent, the video analyzed will poorly recognize the forelimbs. Repeat the exercise for a few times to fine tune the training set in the Cylinder Test configuration being set up.

Appendix G: Ground Truth Reports

G.1 Report compilation procedures

1. Review all video fragments, of 4 minutes (~6,000 frames) each on a frame-by-frame basis;
2. When a wall-rearing activity is observed, note down its start and end frames;
3. Group the wall-rearing activities that are close to each other within 3 seconds into wall-rearing episodes;
4. Record the wall-rearing episodes into ground truth report
5. Sign at the end of the report

G.2 Ground Truth Reports

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX01	01	1	Start	638	627	665									1	2
			End	661	679	674										
		2	Start	897	901										1	1
			End	916	914											
		3	Start	897	901										2	1
			End	916	914											
		4	Start	1,309	1,306	1,326	1,326								3	2
			End	1,321	1,321	1,363	1,353									
		5	Start	1,577	1,593	1,605	1,601	1,621							5	2
			End	1,602	1,598	1,616	1,618	1,632								
		6	Start	1,971	1,982	1,984		1,995							3	2
			End	1,980	2,007	1,992		2,010								
		7	Start	2,471	2,480	2,481	2,486								2	3
			End	2,476	2,484	2,523	2,512									
		8	Start	3,765	3,765	3,779									2	1
			End	3,776	3,790	3,790										
		9	Start	4,247	4,251	4,254									2	1
			End	4,251	4,264	4,261										
		10	Start		4,396											1
			End		4,420											
		11	Start		4,550											1
			End		4,556											
		12	Start		5,741											1
			End		5,781											
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX01	02	1	Start	228	228										1	
			End	240	237											
		2	Start	1,505	1,505	1,528	1,541								3	3
			End	1,523	1,533	1,554	1,551									
		3	Start	2,171	2,176	2,180									2	1
			End	2,176	2,193	2,193										
		4	Start	2,611	2,614	2,620									2	1
			End	2,614	2,632	2,632										
		5	Start	2,838	2,832	2,852	2,845	2,858	2,866						3	3
			End	2,847	2,838	2,890	2,855	2,864	2,887							
		6	Start	3,134	3,134										1	1
			End	3,143	3,145											
		7	Start	4,208	4,208										1	1
			End	4,222	4,223											
		8	Start	4,749	4,749										1	1
			End	4,759	4,758											
		9	Start	5,228	5,228	5,249									2	1
			End	5,246	5,253	5,253										
		10	Start	5,650											1	
			End	5,651												
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX01	03	1	Start	71	71	93										
			End	90	111	107										
		2	Start	1,213	1,184		1,222									
			End	1,232	1,218		1,236									
		3	Start	1,359	1,352		1,361									
			End	1,365	1,359		1,382									
		4	Start	1,740	1,740											
			End	1,768	1,769											
		5	Start		2,515											
			End		2,524											
		6	Start	2,696	2,696											
			End	2,731	2,731											
		7	Start	3,975	3,975	3,985	3,991									
			End	3,982	3,988	4,002	4,002									
		8	Start	5,498	5,496											
			End	5,507	5,507											
		9	Start	5,831	5,831	5,845		5,864								
			End	5,841	5,865	5,862		5,869								
		10	Start	6,100	6,094											
			End	6,107	6,107											
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX02	01	1	Start	1,271	1,271	1,301	1,301								1	1
			End	1,288	1,285	1,322	1,319									
		2	Start	1,796	1,796										1	1
			End	1,805	1,805											
		3	Start	2,110	2,110	2,160	2,162	2,246	2,260						4	3
			End	2,140	2,140	2,180	2,178	2,279	2,269							
		4	Start	2,338	2,338										1	1
			End	2,347	2,341											
		5	Start	2,578	2,578	2,621									1	1
			End	2,588	2,588	2,626										
		6	Start	2,731	2,731	2,777		2,783							1	1
			End	2,774	2,761	2,779		2,847								
		7	Start	3,185	3,185										1	1
			End	3,193	3,193											
		8	Start	4,719	4,719										1	1
			End	4,728	4,728											
		9	Start	5,036	5,036										1	1
			End	5,070	5,068											
		10	Start	5,462	5,462	5,473	5,470	5,497	5,495						1	1
			End	5,471	5,464	5,494	5,492	5,499	5,497							
		11	Start	6,285	6,285										1	1
			End	6,290	6,290											
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX02	02	1	Start	408	408											
			End	416	416											
		2	Start	520	520	546	546									
			End	527	527	561	561									
		3	Start		3,612											
			End		3,641											
		4	Start	3,801	3,801	3,822	3,822									
			End	3,817	3,817	3,873	3,884									
		5	Start	4,269	4,265	4,281	4,281									
			End	4,274	4,274	4,307	4,307									
		6	Start	4,661	4,673	4,674										
			End	4,672	4,704	4,704										
		7	Start	5,259	5,259											
			End	5,275	5,277											
		8	Start		5,951											
			End		5,996											
		9	Start													
			End													
		10	Start													
			End													
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX02	03	1	Start	1,343	1,343										1	1
			End	1,391	1,351											
		2	Start	1,629											1	
			End	1,638												
		3	Start	1,822	1,822										1	1
			End	1,829	1,831											
		4	Start	2,358	2,358	2,400	2,366		2,400						2	3
			End	2,380	2,360	2,409	2,375		2,415							
		5	Start	3,328											1	
			End	3,342												
		6	Start	3,444	3,444	3,465									1	1
			End	3,461	3,486	3,483										
		7	Start	3,706	3,682		3,706								3	2
			End	3,734	3,697		3,734									
		8	Start													
			End													
		9	Start													
			End													
		10	Start													
			End													
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX03	01	1	Start	779												
			End	786												
		2	Start	831	831											
			End	840	865											
		3	Start	926	926											
			End	932	935											
		4	Start	1,054	1,061											
			End	1,080	1,080											
		5	Start		2,864											
			End		2,877											
		6	Start	2,932												
			End	2,941												
		7	Start		3,038											
			End		3,058											
		8	Start	3,482												
			End	3,569												
		9	Start	4,975	4,980											
			End	4,988	4,988											
		10	Start		6,074											
			End		6,084											
		11	Start	6,206	6,191											
			End	6,232	6,232											
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX03	02	1	Start	94	94	100									2	1
			End	96	108	108										
		2	Start		859											1
			End		869											
		3	Start	1,569	1,569	1,582	1,582								2	2
			End	1,580	1,580	1,611	1,611									
		4	Start	1,684	1,684										2	1
			End	1,706	1,706											
		5	Start	1,873	1,873										1	1
			End	1,882	1,882											
		6	Start	1,965	1,965										1	1
			End	1,979	1,979											
		7	Start	2,084												
			End	2,147												
		8	Start	2,484	2,484										1	1
			End	2,506	2,506											
		9	Start	2,630	2,630	2,635	2,635								2	2
			End	2,632	2,632	2,649	2,649									
		10	Start		2,753											1
			End		2,766											
		11	Start	2,850	2,850										1	1
			End	2,865	2,865											
		12	Start	3,157	3,157										2	1
			End	3,172	3,172											
		13	Start	3,339	3,339	3,374	3,370	3,376							1	1
			End	3,350	3,350	3,391	3,373	3,392								
		14	Start	3,489	3,489	3,552	3,552	3,563	3,563						4	5
			End	3,504	3,504	3,555	3,555	3,620	3,620							
		15	Start	3,706	3,706										1	1
			End	3,722	3,722											
		16	Start	3,814	3,814	3,881	3,881								3	3
			End	3,834	3,834	3,898	3,898									
		17	Start	4,595	4,595	4,607	4,604	4,629	4,626	4,654	4,654				4	5
			End	4,604	4,601	4,622	4,622	4,643	4,643	4,675	4,675					
		18	Start	4,804	4,804										2	1
			End	4,839	4,839											
		19	Start	4,922	4,918	4,949	4,924	5,007	4,949	5,012					4	3
			End	4,943	4,922	4,961	4,943	5,030	4,961	5,027						
		20	Start	5,196	5,210	5,218									2	2

Test Subject	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
	21	End	5,210	5,218	5,221											
		Start	5,500	5,498	5,523	5,523	5,554	5,554		5,609					5	7
		End	5,518	5,518	5,542	5,542	5,568	5,568		5,620						
	22	Start	6,136	6,138											2	2
		End	6,150	6,150												

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX03	03	1	Start	192	184										2	2
			End	202	205											
		2	Start	468	468	480	485	491							3	3
			End	473	482	489	520	518								
		3	Start	700	700										1	1
			End	740	743											
		4	Start		1,648											1
			End		1,672											
		5	Start	1,739	1,758										1	1
			End	1,768	1,766											
		6	Start	1,954											1	
			End	1,982												
		7	Start	2,505	2,505		2,523								2	1
			End	2,544	2,511		2,540									
		8	Start	3,029	3,013										1	1
			End	3,072	3,072											
		9	Start	3,296	3,291	3,397	3,392	3,455	3,455						4	5
			End	3,326	3,324	3,409	3,411	3,523	3,523							
		10	Start	3,567	3,563										2	1
			End	3,584	3,584											
		11	Start	3,645	3,640		3,665								1	2
			End	3,685	3,659		3,688									
		12	Start	4,244	4,226		4,311								1	2
			End	4,265	4,267		4,328									
		13	Start	4,404	4,382		4,416								1	2
			End	4,425	4,409		4,425									
		14	Start	4,483	4,487										1	1
			End	4,495	4,495											
		15	Start	4,661	4,598	4,719	4,661	4,781	4,719	4,839	4,783	4,901	4,839		5	6
			End	4,677	4,609	4,753	4,677	4,803	4,753	4,860	4,803	4,917	4,860			
		16	Start	5,068	5,065	5,199	5,199	5,257	5,257						3	4
			End	5,147	5,147	5,219	5,219	5,276	5,276							
		17	Start	5,441	5,445										2	1
			End	5,487	5,484											
		18	Start	5,713	5,713		5,736								1	1
			End	5,722	5,722		5,748									
		19	Start	5,815	5,815										1	1
			End	5,822	5,822											
		20	Start	5,965	5,965										2	2

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
		End	6,018	6,018												
	21	Start	6,179	6,179											1	1
		End	6,205	6,201												

Test Subject	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX04	01	1	Start	75	81	107	102								2	2
			End	103	95	124	127									
		2	Start	1,040	1,040	1,087	1,087								2	2
			End	1,048	1,050	1,112	1,111									
		3	Start	1,445	1,445										1	1
			End	1,460	1,462											
		4	Start	1,689	1,689	1,758	1,765								3	2
			End	1,717	1,711	1,833	1,831									
		5	Start	2,199	2,157		2,206								2	2
			End	2,260	2,198		2,260									
		6	Start	2,743	2,737		2,744								2	2
			End	2,800	2,742		2,800									
		7	Start	3,885	3,922	3,924									2	2
			End	3,921	3,966	3,966										
		8	Start	4,046	4,041		4,050								1	2
			End	4,059	4,048		4,061									
		9	Start	4,591	4,594		4,636								1	4
			End	4,646	4,620		4,646									
		10	Start	4,998	4,998	5,022									2	2
			End	5,018	5,038	5,035										
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX04	02	1	Start	1,933	1,933										1	1
			End	1,950	1,950											
		2	Start	2,142	2,149		2,170		2,185						1	4
			End	2,205	2,160		2,179		2,202							
		3	Start	2,615	2,619	2,624									2	1
			End	2,618	2,638	2,640										
		4	Start	2,903	2,910										1	1
			End	2,933	2,924											
		5	Start	3,233	3,233		3,239		3,298						1	3
			End	3,272	3,235		3,272		3,332							
		6	Start	3,398	3,398	3,415	3,415	3,447	3,476	3,476					4	3
			End	3,407	3,404	3,432	3,428	3,454	3,492	3,492						
		7	Start	4,472	4,472	4,509	4,509								2	2
			End	4,498	4,498	4,529	4,529									
		8	Start	4,817	4,817	4,832	4,832								2	2
			End	4,828	4,828	4,853	4,853									
		9	Start	5,447	5,447										4	2
			End	5,489	5,492											
		10	Start	5,826	5,826	5,855	5,870	5,882	5,882	5,911	5,911	5,960	5,960		5	5
			End	5,851	5,849	5,870	5,879	5,898	5,898	5,946	5,946	5,982	5,982			
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subject	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX04	03	1	Start	2,183	2,179										2	2
			End	2,213	2,203											
		2	Start	2,405	2,421	2,419	2,449	2,446							3	2
			End	2,415	2,446	2,441	2,453	2,449								
		3	Start	2,592	2,589	2,639	2,639	2,751	2,748						5	5
			End	2,631	2,633	2,681	2,681	2,789	2,789							
		4	Start	2,859	2,856	2,882	2,882	2,955	2,955	2,985	2,985				6	6
			End	2,876	2,876	2,890	2,890	2,980	2,980	3,032	3,032					
		5	Start	3,114	3,114										1	1
			End	3,129	3,129											
		6	Start	3,362	3,362		3,402								3	3
			End	3,423	3,398		3,423									
		7	Start	3,620	3,620	3,701	3,701								5	3
			End	3,665	3,665	3,775	3,774									
		8	Start	3,974	3,974	4,053	3,994		4,053						1	2
			End	4,005	3,991	4,059	4,032		4,059							
		9	Start	4,599	4,597										1	2
			End	4,643	4,643											
		10	Start	4,977	4,977										2	2
			End	5,010	5,015											
		11	Start	5,281	5,289	5,293									2	6
			End	5,289	5,320	5,318										
		12	Start	5,605	5,605										2	1
			End	5,643	5,643											
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX06	01	1	Start		159		166		173		180					4
			End		163		171		178		183					
		2	Start	618	604										1	1
			End	620	625											
		3	Start	1,273	1,279	1,285									3	3
			End	1,277	1,309	1,309										
		4	Start	1,530	1,530										1	1
			End	1,546	1,546											
		5	Start	1,706	1,706										2	3
			End	1,759	1,759											
		6	Start	2,209	2,209										1	1
			End	2,223	2,223											
		7	Start	2,447	2,452										2	2
			End	2,465	2,465											
		8	Start	2,895	2,887										1	2
			End	2,926	2,926											
		9	Start	3,158	3,147										1	1
			End	3,172	3,175											
		10	Start	3,607		3,628									1	
			End	3,622		3,637										
		11	Start	3,767											1	
			End	3,773												
		12	Start	3,981	3,975	4,015	3,991		4,015						3	3
			End	4,011	3,981	4,023	4,011		4,023							
		13	Start	4,110	4,110										2	1
			End	4,129	4,129											
		14	Start		4,229										1	4
			End		4,264											
		14	Start	4,342	4,342										1	1
			End	4,352	4,352											
		15	Start	4,617	4,617										2	1
			End	4,639	4,639											
		16	Start	4,788	4,788		4,812								3	4
			End	4,886	4,798		4,886									
		17	Start	5,010	5,016										3	1
			End	5,047	5,045											
		18	Start	5,208	5,203		5,213								1	1
			End	5,233	5,207		5,237									
		19	Start	5,592	5,640		5,648								2	2

Test Subject	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
	20	End	5,672	5,642		5,672										
		Start	5,730	5,723											1	1
		End	5,769	5,770												

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX06	02	21	Start	6,162	6,162										1	2
			End	6,194	6,194											
		1	Start	89	89										1	1
			End	141	141											
		2	Start	991	974		991		1,007						1	3
			End	1,019	984		1,004		1,019							
		3	Start	2,789											1	
			End	2,799												
		4	Start	3,163	3,163										1	1
			End	3,177	3,177											
		5	Start	3,379	3,379	3,391	3,391								1	1
			End	3,387	3,387	3,423	3,420									
		6	Start	3,537	3,542										1	1
			End	3,553	3,553											
		7	Start	4,061	4,056										2	2
			End	4,140	4,111											
		8	Start	5,608	5,601										1	1
			End	5,644	5,650											
		9	Start		5,766											1
			End		5,775											
		10	Start													
			End													
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX06	03	1	Start	2,661	2,708	2,703		2,710							3	3
			End	2,697	2,746	2,707		2,746								
		2	Start	3,231	3,231	3,284	3,278		3,288						3	3
			End	3,282	3,276	3,303	3,285		3,303							
		3	Start	4,807	4,827	4,831									1	1
			End	4,826	4,847	4,851										
		4	Start	4,955	4,945										1	1
			End	4,962	4,968											
		5	Start	5,152	5,152										1	1
			End	5,185	5,180											
		6	Start	5,579	5,523	5,593	5,547		5,584						2	4
			End	5,591	5,545	5,609	5,559		5,609							
		7	Start													
			End													
		8	Start													
			End													
		9	Start													
			End													
		10	Start													
			End													
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Report Prepared by



DAVID CHENG 4/1/2019.

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX01	01	1	Start	635	627	664									1	2
			End	661	679	674										
		2	Start	896	901										1	1
			End	916	914											
		3	Start	897	901										2	1
			End	916	914											
		4	Start	1,309	1,306	1,325	1,326								3	2
			End	1,321	1,321	1,363	1,353									
		5	Start	1,577	1,593	1,605	1,601	1,621							5	2
			End	1,602	1,598	1,615	1,618	1,632								
		6	Start	1,971	1,982	1,984		1,995							3	2
			End	1,980	2,007	1,991		2,010								
		7	Start	2,471	2,480	2,481	2,486								2	3
			End	2,475	2,484	2,523	2,512									
		8	Start	3,765	3,765	3,779									2	1
			End	3,776	3,790	3,791										
		9	Start	4,247	4,251	4,253									2	1
			End	4,250	4,264	4,261										
		10	Start		4,395											1
			End		4,420											
		11	Start		4,550											1
			End		4,555											
		12	Start		5,740											1
			End		5,781											
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX01	02	1	Start	228	228										1	
			End	240	236											
		2	Start	1,504	1,505	1,528	1,541								3	3
			End	1,523	1,533	1,554	1,551									
		3	Start	2,171	2,176	2,180									2	1
			End	2,176	2,192	2,193										
		4	Start	2,611	2,614	2,620									2	1
			End	2,614	2,632	2,631										
		5	Start	2,838	2,832	2,852	2,845		2,858		2,866				3	3
			End	2,847	2,838	2,891	2,855		2,864		2,887					
		6	Start	3,134	3,134										1	1
			End	3,142	3,145											
		7	Start	4,208	4,208										1	1
			End	4,221	4,223											
		8	Start	4,748	4,749										1	1
			End	4,759	4,758											
		9	Start	5,230	5,228	5,249									2	1
			End	5,246	5,253	5,253										
		10	Start	5,650											1	
			End	5,650												
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

YT

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX01	03	1	Start	71	71	93										
		End		90	111	107										
		2	Start	1,212	1,184		1,222									
		End		1,232	1,218		1,236									
		3	Start	1,359	1,351		1,361									
		End		1,365	1,359		1,382									
		4	Start	1,740	1,740											
		End		1,768	1,770											
		5	Start		2,513											
		End			2,524											
		6	Start	2,696	2,696											
		End		2,730	2,731											
		7	Start	3,975	3,975	3,985	3,991									
		End		3,982	3,988	4,002	4,002									
		8	Start	5,497	5,496											
		End		5,507	5,507											
		9	Start	5,832	5,831	5,845		5,864								
		End		5,841	5,865	5,862		5,869								
		10	Start	6,100	6,094											
		End		6,106	6,107											
		11	Start													
		End														
		12	Start													
		End														
		13	Start													
		End														
		14	Start													
		End														
		15	Start													
		End														

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX02	01	1	Start	1,270	1,271	1,301	1,301								1	1
			End	1,288	1,285	1,322	1,319									
		2	Start	1,796	1,796										1	1
			End	1,804	1,805											
		3	Start	2,110	2,111	2,160	2,162								3	2
			End	2,140	2,140	2,180	2,178									
		4	Start	2,246	2,260	2,338	2,338								2	2
			End	2,279	2,269	2,347	2,341									
		5	Start	2,577	2,578	2,621									1	1
			End	2,588	2,588	2,625										
		6	Start	2,730	2,731	2,777		2,782							1	1
			End	2,774	2,761	2,778		2,847								
		7	Start	3,185	3,185										1	1
			End	3,192	3,193											
		8	Start	4,719	4,719										1	1
			End	4,728	4,728											
		9	Start	5,035	5,035										1	1
			End	5,070	5,068											
		10	Start	5,461	5,462	5,473	5,470	5,496	5,495						1	1
			End	5,471	5,464	5,494	5,492	5,499	5,497							
		11	Start	6,285	6,285										1	1
			End	6,290	6,290											
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX02	02	1	Start	408	408											
			End	416	416											
		2	Start	521	520	546	546									
			End	527	527	561	561									
		3	Start		3,611											
			End		3,641											
		4	Start	3,801	3,801	3,822	3,822									
			End	3,816	3,817	3,873	3,884									
		5	Start	4,268	4,265	4,281	4,281									
			End	4,274	4,274	4,307	4,307									
		6	Start	4,661	4,673	4,674										
			End	4,671	4,704	4,704										
		7	Start	5,259	5,258											
			End	5,278	5,277											
		8	Start		5,950											
			End		5,996											
		9	Start													
			End													
		10	Start													
			End													
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX02	03	1	Start	1,342	1,343										1	1
			End	1,391	1,350											
		2	Start	1,629											1	
			End	1,638												
		3	Start	1,822	1,822										1	1
			End	1,828	1,831											
		4	Start	2,358	2,358	2,400	2,366		2,400						2	3
			End	2,380	2,361	2,409	2,375		2,415							
		5	Start	3,327											1	
			End	3,342												
		6	Start	3,445	3,445	3,465									1	1
			End	3,461	3,486	3,483										
		7	Start	3,705	3,682		3,705								3	2
			End	3,734	3,697		3,734									
		8	Start													
			End													
		9	Start													
			End													
		10	Start													
			End													
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX03	01	1	Start	779												
			End	786												
		2	Start	830	830											
			End	840	865											
		3	Start	925	925											
			End	932	935											
		4	Start	1,055	1,062											
			End	1,080	1,080											
		5	Start		2,865											
			End		2,877											
		6	Start	2,931												
			End	2,941												
		7	Start		3,037											
			End		3,058											
		8	Start	3,482												
			End	3,561												
		9	Start	4,975	4,980											
			End	4,988	4,988											
		10	Start		6,074											
			End		6,084											
		11	Start	6,206	6,191											
			End	6,231	6,232											
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX03	02	1	Start	94	94	100									2	1
			End	96	108	108										
		2	Start		859											1
			End		869											
		3	Start	1,568	1,569	1,582	1,582								2	2
			End	1,580	1,580	1,611	1,611									
		4	Start	1,685	1,684										2	1
			End	1,706	1,705											
		5	Start	1,874	1,873										1	1
			End	1,882	1,881											
		6	Start	1,965	1,965										1	1
			End	1,979	1,980											
		7	Start	2,085												
			End	2,147												
		8	Start	2,484	2,484										1	1
			End	2,506	2,506											
		9	Start	2,630	2,630	2,635	2,635								2	2
			End	2,632	2,632	2,649	2,649									
		10	Start		2,753											1
			End		2,765											
		11	Start	2,850	2,850										1	1
			End	2,865	2,865											
		12	Start	3,157	3,157										2	1
			End	3,172	3,172											
		13	Start	3,339	3,339	3,374	3,370	3,376							1	1
			End	3,350	3,350	3,391	3,373	3,392								
		14	Start	3,490	3,490	3,552	3,552	3,564	3,564						4	5
			End	3,504	3,504	3,555	3,555	3,620	3,620							
		15	Start	3,706	3,706										1	1
			End	3,723	3,723											
		16	Start	3,814	3,814	3,881	3,881								3	3
			End	3,834	3,834	3,898	3,898									
		17	Start	4,595	4,595	4,607	4,604	4,629	4,626	4,654	4,654				4	5
			End	4,604	4,601	4,622	4,622	4,643	4,643	4,675	4,675					
		18	Start	4,804	4,804										2	1
			End	4,839	4,839											
		19	Start	4,921	4,918	4,949	4,923	5,007	4,949	5,012					4	3

YT

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
		End	4,943	4,922	4,961	4,943	5,030	4,960	5,027							
	20	Start	5,195	5,210	5,218										2	2
		End	5,210	5,217	5,221											
	21	Start	5,500	5,498	5,524	5,524	5,554	5,554		5,608					5	7
		End	5,518	5,518	5,542	5,542	5,568	5,568		5,620						
	22	Start	6,136	6,138											2	2
		End	6,150	6,150												

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX03	03	1	Start	192	184										2	2
			End	202	205											
		2	Start	468	468	480	485	490							3	3
			End	473	482	489	520	518								
		3	Start	700	700										1	1
			End	740	743											
		4	Start		1,647											1
			End		1,672											
		5	Start	1,738	1,758										1	1
			End	1,768	1,766											
		6	Start	1,954											1	
			End	1,982												
		7	Start	2,505	2,505		2,523								2	1
			End	2,544	2,511		2,540									
		8	Start	3,028	3,013										1	1
			End	3,072	3,072											
		9	Start	3,296	3,291	3,396	3,392	3,455	3,455						4	5
			End	3,326	3,324	3,408	3,411	3,523	3,523							
		10	Start	3,566	3,563										2	1
			End	3,584	3,584											
		11	Start	3,645	3,640		3,665								1	2
			End	3,685	3,659		3,688									
		12	Start	4,244	4,225		4,311								1	2
			End	4,265	4,267		4,328									
		13	Start	4,404	4,382		4,416								1	2
			End	4,425	4,409		4,425									
		14	Start	4,483	4,487										1	1
			End	4,495	4,495											
		15	Start	4,661	4,598	4,719	4,661	4,781	4,719	4,839	4,783	4,901	4,839	4,901	5	6
			End	4,677	4,609	4,753	4,677	4,803	4,753	4,860	4,803	4,917	4,860	4,919		
		16	Start	5,068	5,065	5,199	5,199	5,257	5,257						3	4
			End	5,147	5,147	5,219	5,219	5,276	5,276							
		17	Start	5,440	5,445										2	1
			End	5,487	5,484											
		18	Start	5,713	5,713		5,736								1	1
			End	5,722	5,722		5,748									
		19	Start	5,815	5,815										1	1

YT

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
		End	5,822	5,822												
		20 Start	5,965	5,965											2	2
	21	End	6,018	6,018												
		Start	6,179	6,179											1	1
		End	6,205	6,201												

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX04	01	1	Start	75	81	107	102								2	2
			End	103	95	124	127									
		2	Start	1,040	1,040	1,087	1,087								2	2
			End	1,048	1,050	1,112	1,111									
		3	Start	1,445	1,445										1	1
			End	1,460	1,462											
		4	Start	1,689	1,689	1,758	1,765								3	2
			End	1,717	1,711	1,833	1,831									
		5	Start	2,198	2,157		2,206								2	2
			End	2,260	2,197		2,260									
		6	Start	2,743	2,737		2,744								2	2
			End	2,800	2,742		2,800									
		7	Start	3,885	3,921	3,924									2	2
			End	3,921	3,965	3,966										
		8	Start	4,046	4,041		4,050								1	2
			End	4,059	4,047		4,061									
		9	Start	4,590	4,594		4,636								1	4
			End	4,646	4,620		4,646									
		10	Start	4,998	4,998	5,021									2	2
			End	5,018	5,038	5,035										
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX04	02	1	Start	1,932	1,932										1	1
			End	1,950	1,950											
		2	Start	2,142	2,149		2,170		2,185						1	4
			End	2,205	2,160		2,180		2,200							
		3	Start	2,615	2,618	2,624									2	1
			End	2,618	2,638	2,640										
		4	Start	2,903	2,910										1	1
			End	2,933	2,924											
		5	Start	3,233	3,233		3,238		3,298						1	3
			End	3,272	3,234		3,270		3,332							
		6	Start	3,398	3,398	3,415	3,415	3,447	3,476	3,476					4	3
			End	3,407	3,405	3,432	3,427	3,454	3,492	3,492						
		7	Start	4,472	4,472	4,509	4,530								2	2
			End	4,498	4,498	4,529	4,529									
		8	Start	4,817	4,817	4,832	4,832								2	2
			End	4,828	4,828	4,853	4,853									
		9	Start	5,447	5,447										4	2
			End	5,489	5,491											
		10	Start	5,826	5,826	5,855	5,870	5,882	5,882	5,911	5,911	5,960	5,960		5	5
			End	5,851	5,849	5,870	5,880	5,898	5,900	5,946	5,946	5,982	5,982			
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

YT

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX04	03	1	Start	2,183	2,179										2	2
		End		2,213	2,200											
		2	Start	2,405	2,421	2,419	2,449	2,446							3	2
		End		2,415	2,446	2,441	2,452	2,449								
		3	Start	2,592	2,589	2,639	2,639	2,751	2,748						5	5
		End		2,631	2,633	2,681	2,681	2,789	2,790							
		4	Start	2,859	2,856	2,882	2,882	2,955	2,955	2,985	2,985				6	6
		End		2,876	2,875	2,890	2,890	2,980	2,980	3,032	3,032					
		5	Start	3,114	3,114										1	1
		End		3,129	3,129											
		6	Start	3,362	3,362		3,402								3	3
		End		3,423	3,400		3,423									
		7	Start	3,620	3,620	3,701	3,701								5	3
		End		3,665	3,665	3,775	3,774									
		8	Start	3,974	3,974	4,053	3,994	4,053							1	2
		End		4,005	3,991	4,059	4,032	4,059								
		9	Start	4,599	4,597										1	2
		End		4,643	4,643											
		10	Start	4,977	4,977										2	2
		End		5,010	5,015											
		11	Start	5,281	5,289	5,293									2	6
		End		5,289	5,320	5,318										
		12	Start	5,605	5,604										2	1
		End		5,643	5,643											
		13	Start													
		End														
		14	Start													
		End														
		15	Start													
		End														

YT

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
		End	5,232	5,207		5,237										
	19	Start	5,592	5,640		5,648									2	2
		End	5,672	5,642		5,672										
	20	Start	5,730	5,723											1	1
		End	5,768	5,770												
	21	Start	6,162	6,162											1	2
		End	6,194	6,194												

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX06	01	1	Start													
			End	159		166		173		180						4
		2	Start	618	604										1	1
			End	620	625											
		3	Start	1,273	1,279	1,285									3	3
			End	1,277	1,309	1,309										
		4	Start	1,530	1,530										1	1
			End	1,546	1,546											
		5	Start	1,706	1,706										2	3
			End	1,759	1,758											
		6	Start	2,209	2,209										1	1
			End	2,223	2,223											
		7	Start	2,447	2,452										2	2
			End	2,465	2,465											
		8	Start	2,895	2,887										1	2
			End	2,927	2,926											
		9	Start	3,158	3,147										1	1
			End	3,172	3,175											
		10	Start	3,607		3,628									1	
			End	3,622		3,637										
		11	Start	3,767											1	
			End	3,773												
		12	Start	3,981	3,975	4,015	3,991	4,015							3	3
			End	4,010	3,981	4,022	4,011	4,023								
		13	Start	4,110	4,110										2	1
			End	4,129	4,129											
		14	Start		4,229										1	4
			End		4,263											
		14	Start	4,342	4,342										1	1
			End	4,352	4,352											
		15	Start	4,617	4,617										2	1
			End	4,639	4,639											
		16	Start	4,788	4,788		4,812								3	4
			End	4,886	4,800		4,886									
		17	Start	5,010	5,016										3	1
			End	5,047	5,045											
		18	Start	5,208	5,203		5,213								1	1

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX06	02	1	Start	89	89										1	1
			End	141	141											
		2	Start	991	974		991		1,007						1	3
			End	1,019	983		1,002		1,019							
		3	Start	2,789											1	
			End	2,799												
		4	Start	3,163	3,163										1	1
			End	3,177	3,177											
		5	Start	3,379	3,379	3,390	3,391								1	1
			End	3,387	3,387	3,423	3,420									
		6	Start	3,537	3,542										1	1
			End	3,553	3,553											
		7	Start	4,061	4,056										2	2
			End	4,110	4,111											
		8	Start	5,608	5,601										1	1
			End	5,645	5,650											
		9	Start		5,766											1
			End		5,775											
		10	Start													
			End													
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

YT

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX06	03	1	Start	2,661	2,708	2,703		2,710							3	3
			End	2,697	2,746	2,707		2,746								
		2	Start	3,231	3,231	3,284	3,278	3,288							3	3
			End	3,282	3,277	3,303	3,286	3,303								
		3	Start	4,807	4,827	4,831									1	1
			End	4,826	4,846	4,851										
		4	Start	4,955	4,945										1	1
			End	4,962	4,968											
		5	Start	5,152	5,152										1	1
			End	5,185	5,180											
		6	Start	5,579	5,522	5,593	5,546	5,584							2	4
			End	5,591	5,545	5,609	5,560	5,609								
		7	Start													
			End													
		8	Start													
			End													
		9	Start													
			End													
		10	Start													
			End													
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Report Prepared by

YUAN TIAN

Name

Graduate Student

Joanne Matsubara's lab April 2nd, 2019

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX01	01	1	Start	635	627	664									1	2
			End	661	679	674										
		2	Start	896	901										1	1
			End	916	914											
		3	Start	897	901										2	1
			End	916	914											
		4	Start	1,309	1,306	1,325	1,326								3	2
			End	1,320	1,320	1,363	1,353									
		5	Start	1,577	1,593	1,605	1,601	1,621							5	2
			End	1,602	1,599	1,615	1,619	1,632								
		6	Start	1,971	1,982	1,984		1,995							3	2
			End	1,980	2,007	1,991		2,010								
		7	Start	2,471	2,480	2,481	2,486								2	3
			End	2,475	2,484	2,523	2,512									
		8	Start	3,765	3,765	3,780									2	1
			End	3,775	3,790	3,791										
		9	Start	4,247	4,251	4,253									2	1
			End	4,250	4,264	4,261										
		10	Start		4,395											1
			End		4,420											
		11	Start		4,550											1
			End		4,555											
		12	Start		5,740											1
			End		5,781											
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

[illegible]

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX01	03	1	Start	71	71	93										
			End	90	111	107										
		2	Start	1,212	1,184		1,222									
			End	1,232	1,217		1,236									
		3	Start	1,359	1,351		1,361									
			End	1,365	1,358		1,382									
		4	Start	1,740	1,740											
			End	1,768	1,770											
		5	Start		2,513											
			End		2,524											
		6	Start	2,696	2,696											
			End	2,730	2,731											
		7	Start	3,975	3,975	3,985	3,991									
			End	3,982	3,989	4,002	4,002									
		8	Start	5,497	5,496											
			End	5,507	5,507											
		9	Start	5,832	5,831	5,845		5,864								
			End	5,841	5,865	5,861		5,869								
		10	Start	6,100	6,094											
			End	6,106	6,107											
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX02	02	1	Start	408	408											
		End	416	416												
		2	Start	521	520	546	546									
		End	526	527	561	561										
		3	Start		3,611											
		End		3,641												
		4	Start	3,801	3,801	3,822	3,822									
		End	3,815	3,817	3,873	3,884										
		5	Start	4,268	4,265	4,281	4,281									
		End	4,275	4,275	4,307	4,307										
		6	Start	4,661	4,673	4,674										
		End	4,671	4,704	4,704											
		7	Start	5,259	5,258											
		End	5,278	5,277												
		8	Start		5,950											
		End		5,996												
		9	Start													
		End														
		10	Start													
		End														
		11	Start													
		End														
		12	Start													
		End														
		13	Start													
		End														
		14	Start													
		End														
		15	Start													
		End														

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX03	01	1	Start	779												
			End	785												
		2	Start	830	830											
			End	840	865											
		3	Start	925	925											
			End	932	935											
		4	Start	1,055	1,062											
			End	1,080	1,080											
		5	Start		2,865											
			End		2,877											
		6	Start	2,931												
			End	2,940												
		7	Start		3,037											
			End		3,058											
		8	Start	3,482												
			End	3,560												
		9	Start	4,975	4,980											
			End	4,988	4,988											
		10	Start		6,074											
			End		6,084											
		11	Start	6,206	6,191											
			End	6,231	6,232											
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX03	02	1	Start	94	94	100									2	1
			End	96	108	108										
		2	Start		859											1
			End		869											
		3	Start	1,568	1,569	1,583	1,583								2	2
			End	1,580	1,580	1,611	1,611									
		4	Start	1,685	1,684										2	1
			End	1,706	1,705											
		5	Start	1,874	1,873										1	1
			End	1,882	1,881											
		6	Start	1,965	1,965										1	1
			End	1,980	1,980											
		7	Start	2,085												
			End	2,147												
		8	Start	2,484	2,484										1	1
			End	2,506	2,506											
		9	Start	2,630	2,630	2,635	2,635								2	2
			End	2,632	2,632	2,649	2,649									
		10	Start		2,753											1
			End		2,765											
		11	Start	2,850	2,850										1	1
			End	2,865	2,865											
		12	Start	3,157	3,157										2	1
			End	3,172	3,172											
		13	Start	3,339	3,339	3,374	3,370	3,376							1	1
			End	3,350	3,350	3,391	3,373	3,392								
		14	Start	3,490	3,490	3,552	3,552	3,564	3,565						4	5
			End	3,504	3,505	3,555	3,555	3,620	3,620							
		15	Start	3,706	3,706										1	1
			End	3,723	3,723											
		16	Start	3,814	3,814	3,881	3,881								3	3
			End	3,834	3,834	3,898	3,898									
		17	Start	4,595	4,595	4,607	4,604	4,629	4,626	4,654	4,654				4	5
			End	4,604	4,601	4,622	4,622	4,643	4,643	4,675	4,675					
		18	Start	4,804	4,804										2	1
			End	4,839	4,839											
		19	Start	4,921	4,918	4,949	4,923	5,007	4,949	5,012					4	3

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
		End	4,943	4,922	4,961	4,943	5,030	4,960	5,027							
	20	Start	5,195	5,210	5,218										2	2
		End	5,210	5,217	5,221											
	21	Start	5,500	5,498	5,524	5,524	5,554	5,554		5,608					5	7
		End	5,518	5,518	5,542	5,542	5,568	5,568		5,620						
	22	Start	6,136	6,138											2	2
		End	6,150	6,150												

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
	20	End	5,822	5,822												
		Start	5,965	5,965											2	2
		End	6,018	6,018												
	21	Start	6,179	6,179											1	1
		End	6,205	6,205												

Test Subj - Session Episode			1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX04	02	1	Start	1,932	1,932										1	1
		End		1,950	1,950											
		2	Start	2,142	2,149		2,170		2,185						1	4
		End		2,205	2,160		2,180		2,200							
		3	Start	2,615	2,618	2,624									2	1
		End		2,618	2,638	2,640										
		4	Start	2,903	2,910										1	1
		End		2,933	2,924											
		5	Start	3,233	3,233		3,238		3,298						1	3
		End		3,272	3,233		3,270		3,332							
		6	Start	3,398	3,398	3,415	3,415	3,447	3,476	3,476					4	3
		End		3,407	3,405	3,432	3,427	3,454	3,492	3,492						
		7	Start	4,472	4,472	4,509	4,530								2	2
		End		4,498	4,500	4,529	4,529									
		8	Start	4,817	4,817	4,832	4,832								2	2
		End		4,828	4,828	4,853	4,853									
		9	Start	5,447	5,447										4	2
		End		5,489	5,491											
		10	Start	5,826	5,826	5,855	5,870	5,882	5,882	5,911	5,911	5,960	5,960		5	5
		End		5,851	5,850	5,870	5,880	5,898	5,900	5,946	5,946	5,982	5,982			
		11	Start													
		End														
		12	Start													
		End														
		13	Start													
		End														
		14	Start													
		End														
		15	Start													
		End														

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX06	01	1	Start	159		166		173		180						4
		End		163		171		178		183						
		2	Start	618	604										1	1
		End	620	625												
		3	Start	1,273	1,279	1,285									3	3
		End	1,277	1,309	1,309											
		4	Start	1,530	1,530										1	1
		End	1,546	1,546												
		5	Start	1,706	1,706										2	3
		End	1,759	1,758												
		6	Start	2,209	2,209										1	1
		End	2,223	2,223												
		7	Start	2,447	2,452										2	2
		End	2,465	2,465												
		8	Start	2,895	2,887										1	2
		End	2,927	2,926												
		9	Start	3,158	3,147										1	1
		End	3,172	3,175												
		10	Start	3,607		3,628									1	
		End	3,622		3,637											
		11	Start	3,767											1	
		End	3,773													
		12	Start	3,981	3,975	4,015	3,991	4,015							3	3
		End	4,010	3,980	4,022	4,011	4,023									
		13	Start	4,110	4,110										2	1
		End	4,129	4,129												
		14	Start		4,229										1	4
		End		4,263												
		14	Start	4,342	4,342										1	1
		End	4,352	4,352												
		15	Start	4,617	4,617										2	1
		End	4,639	4,639												
		16	Start	4,788	4,788		4,812								3	4
		End	4,886	4,800		4,886										
		17	Start	5,010	5,016										3	1
		End	5,047	5,045												
		18	Start	5,208	5,203		5,213								1	1

104

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX06	02	1	Start	89	89										1	1
			End	141	141											
		2	Start	991	974		991		1,007						1	3
			End	1,019	983		1,002		1,019							
		3	Start	2,789											1	
			End	2,799												
		4	Start	3,163	3,163										1	1
			End	3,177	3,177											
		5	Start	3,379	3,379	3,390	3,391								1	1
			End	3,387	3,387	3,423	3,420									
		6	Start	3,537	3,542										1	1
			End	3,553	3,553											
		7	Start	4,061	4,056										2	2
			End	4,110	4,111											
		8	Start	5,608	5,600										1	1
			End	5,645	5,650											
		9	Start		5,765											1
			End		5,775											
		10	Start													
			End													
		11	Start													
			End													
		12	Start													
			End													
		13	Start													
			End													
		14	Start													
			End													
		15	Start													
			End													

Test Subj	Session	Episode	1st Touch		2nd Touch		3rd Touch		4th Touch		5th Touch		6th Touch		Total Touches in Episode	
			Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
EMX06	03	1	Start	2,661	2,708	2,703		2,710							3	3
		End		2,697	2,746	2,707		2,746								
		2	Start	3,231	3,231	3,284	3,278		3,288						3	3
		End		3,282	3,277	3,303	3,286		3,303							
		3	Start	4,807	4,827	4,831									1	1
		End		4,826	4,845	4,851										
		4	Start	4,955	4,945										1	1
		End		4,962	4,968											
		5	Start	5,152	5,152										1	1
		End		5,185	5,180											
		6	Start	5,579	5,522	5,593	5,546		5,584						2	4
		End		5,591	5,545	5,609	5,560		5,609							
		7	Start													
		End														
		8	Start													
		End														
		9	Start													
		End														
		10	Start													
		End														
		11	Start													
		End														
		12	Start													
		End														
		13	Start													
		End														
		14	Start													
		End														
		15	Start													
		End														

Report Prepared by

WONG DIK HWT

Name

Appendix H: Matlab scripts

```

%Prepare Input Batch file
inputfile = "G:\Tim's Lab\EMXMas\PalmGridBatch1.xlsx";

delimiter=' ';
PalmGridBatch_struct =importdata(inputfile,delimiter,1);

Cylinder = PalmGridBatch_struct.data.Sheet1;
No_of_entries = size(Cylinder,1);
outputfile = "G:\Tim's Lab\EMXMas\PalmGridParam.xlsx";

for i=1:No_of_entries
    xlswrite(outputfile,PalmGridBatch_struct.textdata.Sheet1(i,2), "PalmGridSheet", "C2");
    xlswrite(outputfile,PalmGridBatch_struct.textdata.Sheet1(i,3), "PalmGridSheet", "C3");
    xlswrite(outputfile,PalmGridBatch_struct.textdata.Sheet1(i,4), "PalmGridSheet", "C4");
    xlswrite(outputfile,PalmGridBatch_struct.data.Sheet1(i,1), "PalmGridSheet", "C5");
    xlswrite(outputfile,PalmGridBatch_struct.data.Sheet1(i,2), "PalmGridSheet", "C6");
    xlswrite(outputfile,PalmGridBatch_struct.data.Sheet1(i,3), "PalmGridSheet", "C7");

    string = [ "Start Processing File: ", PalmGridBatch_struct.textdata.Sheet1(i,1)];
    disp(string);

    FunctionPalm_LeftRefine();
    FunctionPalm_RightRefine();
    string = [ "Complete execution of file: ", PalmGridBatch_struct.textdata.Sheet1(i,1)];
    disp(string);
    string = ( "_____");
    disp(string);

end
clear

```



```

%Parameter Script
function FunctionPalm_LeftRefine()

%***** Read PalmGrid Parameter Files *****%

inputfile = "G:\Tim's Lab\EMXMas\PalmGridParam.xlsx";

delimiter=' ';
PalmGridParam_struct =importdata(inputfile,delimiter,1);

%***** Step 1: Input Files Specifications *****
pc_cutoff = 0.1; % cutoff probability of predicted coordinates
Frame_Rate = 25; %Video recorded in 25 frames per second
Center_X = PalmGridParam_struct.data.PalmGridSheet(1,1); %Cartesian Coordinates of Center (X,Y) of
Cylinder
Center_Y = PalmGridParam_struct.data.PalmGridSheet(2,1);
Inner_Diameter = PalmGridParam_struct.data.PalmGridSheet(3,1); %Inner diameter of cylinder in pixels

filename = char(PalmGridParam_struct.textdata.PalmGridSheet(1,1));
video = char(PalmGridParam_struct.textdata.PalmGridSheet(2,1));

%***** Step 2: Smoothing Specifications *****
LabeledPointsPerPaw = 6; %Number of labeled points per paw
Smooth_window = 24; %Size of time window for initial signal smoothing

%***** Step 3: Left / Right Paw Polar Conversion *****
Peak_Threshold = 3; %All data within 2 pixels of local peaks are excluded

%***** Step 4: Extract Congruent Zones Specifications *****
TimeWindow = 6; %Specify time window of smoothing
Threshold = 5; %Specify threshold of screening within 5 pixels
No_of_labeled_points = 6; %Specify number of labeled points per paw
Min_Slow_Movement_Frames = 3; %Minimal Number of consecutive slow movement frames that paw shall be
held before considering a contact
Min_Flag_To_Stay = 0; %Minimal Number of stable points detected in any particular frame to be
considered in Congruence Zone
area_statistics_columns = 13;

%***** Step 5: Extract Coherent Subfragment Specifications *****
Max_Palm_Threshold = 8; %Palm that moves beyond 8 pixels per frame shall be considered fast movement
Min_Palm_Threshold = 5; %Palm that moves below 5 pixels per frame shall be considered slow movement
Stationary_Mvt_Threshold = 1; %Average Movement of less than 1 pixels per frame is considered
stationary movements
safety_margin = 0.85; %Safety margin of Inner Diameter, within which paw most likely not wall rearing

Min_Congruence = 2; %MC=2; Minimum number of congruence points required to count as congruent intervals
Max_SubFragment_Gap = 2; %Maximum number of frames that two subfragments can be separated in order to
be considered merging two subfragments as one
Congruence_window = 8; %CW = 6; Number of frames (hence time windows) within which minima of FingerTips
- Palm are coherently reached
Inspection_window = 12; %IW = 12; Time Window within which congruence shall be inspected

```

```

Columns_of_SubFragments = 11;

%***** Step 6: Extract Congruent Points Specifications *****

%***** Step 7: Decisions Evaluation Specifications *****
Fragment_ref = 0;
Crawling_low_threshold = 10;
Crawling_high_threshold = 100;
Minimum_touch_duration = Frame_Rate * 0.1; % At least 0.1 seconds of slow movements to classify as ↙
stable touch

Min_stat_computation = 3; %Minimum time frames below which repartitioned subfragment will not be ↙
computed of statistics

Variation_TH = 13;
Variation_High_TH = 100;
Max_radii_differentials = 120;
Inner_radius_outlier_removal_factor = 1.0;

Transition_TH = 100; %change in number of pixels between frame exceed 100 as transition threshold

%***** Step 8: Consolidate subfragments into episodes *****
%Time difference within which subfragments consolidate
Consolidate_SubFrag_time = 3;
%Minimum Episode Time Parameter refers to the least possible episode
%considered for reporting
Min_Episode_Time = 0.25;

%***** Step 9: Output Files Specifications *****
outputfile = char(PalmGridParam_struct.txtdata.PalmGridSheet(3,1));

%***** Start Main Program *****
%***** Input Files Specifications *****
%Process Left Paw
Process_left_paw = 1;

RawTable=csvread(filename,3,1);
[no_of_rows, columns]=size(RawTable);

%Initialize Left Front Paw and Right Front Paw Arrays
Lfp = zeros(no_of_rows, 12);
Lfp_P = zeros(no_of_rows, 7);
Rfp = zeros(no_of_rows, 12);
Rfp_P = zeros(no_of_rows, 7);

for i=1:no_of_rows %Assign RawTable to Tables Head/Lfp/Rfp/Lbp/Rbp, and evaluate Likelihood
%Lfp1 = Left Front Paw's Thumb

```

```

%Lfp2 = Left Front Paw's Index finger
%Lfp3 = Left Front Paw's Long finger
%Lfp4 = Left Front Paw's Ring finger
%Lfp5 = Left Front Paw's Small finger
%Lfp6 = Left Front Paw's Radius

Lfp(i,1)=RawTable(i,1); %Lfp1_x
Lfp(i,2)=RawTable(i,2); %Lfp1_y
Lfp(i,3)=RawTable(i,4); %Lfp2_x
Lfp(i,4)=RawTable(i,5); %Lfp2_y
Lfp(i,5)=RawTable(i,7); %Lfp3_x
Lfp(i,6)=RawTable(i,8); %Lfp3_y
Lfp(i,7)=RawTable(i,10); %Lfp4_x
Lfp(i,8)=RawTable(i,11); %Lfp4_y
Lfp(i,9)=RawTable(i,13); %Lfp5_x
Lfp(i,10)=RawTable(i,14); %Lfp5_y
Lfp(i,11)=RawTable(i,16); %Lfp6_x
Lfp(i,12)=RawTable(i,17); %Lfp6_y

Lfp_P(i,1)=RawTable(i,3); %Likelihood Lfp1
Lfp_P(i,2)=RawTable(i,6); %Likelihood Lfp2
Lfp_P(i,3)=RawTable(i,9); %Likelihood Lfp3
Lfp_P(i,4)=RawTable(i,12); %Likelihood Lfp4
Lfp_P(i,5)=RawTable(i,15); %Likelihood Lfp5
Lfp_P(i,6)=RawTable(i,18); %Likelihood Lfp6

Reliability = 6; %Compute Reliability of Coordinates in Head
for j=1:6
    if Lfp_P(i,j) < pc_cutoff
        Reliability=Reliability-1;
    end
end
Lfp_P(i,7)=Reliability;

%Rfp1 = Right Front Paw's Thumb
%Rfp2 = Right Front Paw's Index finger
%Rfp3 = Right Front Paw's Long finger
%Rfp4 = Right Front Paw's Ring finger
%Rfp5 = Right Front Paw's Small finger
%Rfp6 = Right Front Paw's Radius

Rfp(i,1)=RawTable(i,19); %Rfp1_x
Rfp(i,2)=RawTable(i,20); %Rfp1_y
Rfp(i,3)=RawTable(i,22); %Rfp2_x
Rfp(i,4)=RawTable(i,23); %Rfp2_y
Rfp(i,5)=RawTable(i,25); %Rfp3_x
Rfp(i,6)=RawTable(i,26); %Rfp3_y
Rfp(i,7)=RawTable(i,28); %Rfp4_x
Rfp(i,8)=RawTable(i,29); %Rfp4_y
Rfp(i,9)=RawTable(i,31); %Rfp3_x
Rfp(i,10)=RawTable(i,32); %Rfp3_y
Rfp(i,11)=RawTable(i,34); %Rfp4_x

```

```

Rfp(i,12)=RawTable(i,35); %Rfp4_y

Rfp_P(i,1)=RawTable(i,21); %Likelihood Rfp_1
Rfp_P(i,2)=RawTable(i,24); %Likelihood Rfp_2
Rfp_P(i,3)=RawTable(i,27); %Likelihood Rfp_3
Rfp_P(i,4)=RawTable(i,30); %Likelihood Rfp_4
Rfp_P(i,5)=RawTable(i,33); %Likelihood Rfp_3
Rfp_P(i,6)=RawTable(i,36); %Likelihood Rfp_4

Reliability = 6; %Compute Reliability of Coordinates in Left Front Paw
for j=1:6
    if Rfp_P(i,j) < pc_cutoff
        Reliability=Reliability-1;
    end
end
Rfp_P(i,7)=Reliability;

end

%***** Smoothing Specifications *****
%Smooth the predicted coordinates by averaging +/- 12 extracted coordinates
half_Smooth_window = Smooth_window / 2;

buffer = zeros(no_of_rows, 2*LabeledPointsPerPaw);
for i = 1:half_Smooth_window %Smooth Left Front Paw for first few no_of_rows
    for j= 1:2*LabeledPointsPerPaw
        buffer (i,j) = Lfp (i,j);
    end
end
for i = half_Smooth_window:(no_of_rows-half_Smooth_window) %Beyond the first few no_of_rows, Lfp ✓
    coordinates are smoothed based on past Smoothed_window records
    for j= 1:2*LabeledPointsPerPaw
        Avg_coor = 0.00;
        for k = 0:half_Smooth_window
            Avg_coor = Avg_coor + (Lfp (i-k+1,j) + Lfp(i+k,j))/((k+1)/2);
        end
        buffer (i,j) = Avg_coor / Smooth_window;
    end
end
for i = (no_of_rows-half_Smooth_window):no_of_rows %Smooth Left Front Paw for last few no_of_rows
    for j= 1:2*LabeledPointsPerPaw
        buffer (i,j) = Lfp (i,j);
    end
end
filteredLfp = buffer;

for i = 1:half_Smooth_window %Smooth Right Front Paw for first few no_of_rows
    for j= 1:2*LabeledPointsPerPaw
        buffer (i,j) = Rfp (i,j);
    end
end
end

```



```

for i = half_Smooth_window:(no_of_rows-half_Smooth_window) %Beyond the first few no_of_rows, Rfp ✓
coordinates are smoothed based on past Smoothed_window records
    for j= 1:2*LabeledPointsPerPaw
        Avg_coor = 0.00;
        for k = 0:half_Smooth_window
            Avg_coor = Avg_coor + (Rfp (i-k+1,j) + Rfp(i+k,j))/((k+1)/2);
        end
        buffer (i,j) = Avg_coor / Smooth_window;
    end
end
for i = (no_of_rows-half_Smooth_window):no_of_rows %Smooth Right Front Paw for last few no_of_rows
    for j= 1:2*LabeledPointsPerPaw
        buffer (i,j) = Rfp (i,j);
    end
end
filteredRfp = buffer;

%***** Left / Right Paw Polar Conversion *****

Lfp_Polar_Radius = zeros(no_of_rows,6);
Lfp_Polar_Angles = zeros(no_of_rows,6);

Inspect_Range = floor(Frame_Rate/4);
Minimum_Inspection_Window = 6;

for i=1:no_of_rows

    %Translate five Left Front Paw fingertips' cartesian to polar coordinates
    [Lfp_Polar_Angles(i,1),Lfp_Polar_Radius(i,1)] = cart2pol(filteredLfp(i,1)-Center_X,filteredLfp(i,2)- ✓
Center_Y);
    [Lfp_Polar_Angles(i,2),Lfp_Polar_Radius(i,2)] = cart2pol(filteredLfp(i,3)-Center_X,filteredLfp(i,4)- ✓
Center_Y);
    [Lfp_Polar_Angles(i,3),Lfp_Polar_Radius(i,3)] = cart2pol(filteredLfp(i,5)-Center_X,filteredLfp(i,6)- ✓
Center_Y);
    [Lfp_Polar_Angles(i,4),Lfp_Polar_Radius(i,4)] = cart2pol(filteredLfp(i,7)-Center_X,filteredLfp(i,8)- ✓
Center_Y);
    [Lfp_Polar_Angles(i,5),Lfp_Polar_Radius(i,5)] = cart2pol(filteredLfp(i,9)-Center_X,filteredLfp(i,10)- ✓
Center_Y);
    [Lfp_Polar_Angles(i,6),Lfp_Polar_Radius(i,6)] = cart2pol(filteredLfp(i,11)-Center_X,filteredLfp(i,12)- ✓
-Center_Y);

end

Polar_Radius = Lfp_Polar_Radius;

Lfp_Polar_Radius = zeros(no_of_rows,6);
Lfp_Polar_Angles = zeros(no_of_rows,6);

for i=1:no_of_rows

```

```

    %Translate five Left Front Paw fingertips' cartesian to polar coordinates
    [Lfp_Polar_Angles(i,1),Lfp_Polar_Radius(i,1)]= cart2pol(Lfp(i,1)-Center_X,Lfp(i,2)-Center_Y);
    [Lfp_Polar_Angles(i,2),Lfp_Polar_Radius(i,2)]= cart2pol(Lfp(i,3)-Center_X,Lfp(i,4)-Center_Y);
    [Lfp_Polar_Angles(i,3),Lfp_Polar_Radius(i,3)]= cart2pol(Lfp(i,5)-Center_X,Lfp(i,6)-Center_Y);
    [Lfp_Polar_Angles(i,4),Lfp_Polar_Radius(i,4)]= cart2pol(Lfp(i,7)-Center_X,Lfp(i,8)-Center_Y);
    [Lfp_Polar_Angles(i,5),Lfp_Polar_Radius(i,5)]= cart2pol(Lfp(i,9)-Center_X,Lfp(i,10)-Center_Y);
    [Lfp_Polar_Angles(i,6),Lfp_Polar_Radius(i,6)]= cart2pol(Lfp(i,11)-Center_X,Lfp(i,12)-Center_Y);

end

Raw_Polar_Radius = Lfp_Polar_Radius;

%***** Extract Congruent Zones Specifications *****
%Initialize working arrays
Average_Polar_Radius = zeros(no_of_rows, No_of_labeled_points*2);
Congruence_row = zeros(no_of_rows,1);

%Average Radial Displacements of each labeled points to extract principal
%axes
Average_Factor = 2 * TimeWindow + 1;
for i= (TimeWindow + 1): (no_of_rows - TimeWindow)
    for j=1:6
        Average_Pr = 0;
        for k = (i-TimeWindow):(i+TimeWindow)
            Average_Pr = Average_Pr + Polar_Radius(k,j);
        end
        Average_Pr = Average_Pr / Average_Factor;
        Average_Polar_Radius(i,j) = Average_Pr;
        Average_Polar_Radius(i,j+No_of_labeled_points) = Average_Polar_Radius(i,j) - ✓
    end
end

%Filter no_of_rows whose average radial displacement of either labeled points are
%beyond Threshold
Average_Polar_Radius_templ = Average_Polar_Radius;
for i= (TimeWindow + 1): (no_of_rows - TimeWindow)
    Flag_to_stay = 0; % Inspect all ?radius within threshold
    for j=No_of_labeled_points+1:No_of_labeled_points*2
        if (abs(Average_Polar_Radius_templ(i,j)) < Threshold && abs(Average_Polar_Radius_templ(i,j)) > ✓
0)
            %If any labeled points difference from last labeled points <
            %Threshold, flag the row to stay
            Flag_to_stay = 1;
        end
    end
    %If all labeled points radial displacement > Threshold, paw is likely
    %moving => unlikely to touch wall / very slow locomotion / resting on
    %floor
    if Flag_to_stay == 0

```



```

        for j=1:No_of_labeled_points * 2
            Average_Polar_Radius_temp1(i,j)=0;
        end
    else
        Congruence_row (i) = 1;
    end
end
end

%Filter Congruence no_of_rows for at least 2 out of 5 finger tips demonstrate
%slow movements
Average_Polar_Radius_temp2=Average_Polar_Radius_temp1;
for i=1:no_of_rows
    if Congruence_row(i) > 0
        Flag_to_stay = 0;
        for j= (No_of_labeled_points + 1):(No_of_labeled_points*2 - 1)
            if abs(Average_Polar_Radius_temp2(i,j)) < Threshold && abs(Average_Polar_Radius_temp2(i,j)) <
                > 0
                %Count number of finger tips that demonstrate slow movements
                %between frames
                Flag_to_stay = Flag_to_stay + 1;
            end
        end
        if Flag_to_stay < Min_Flag_To_Stay
            %if 1 out of 5 fingertips demonstrate movement < (Threshold; currently = 5)
            %pixels between successive frames; the frame can stay
            for j=1:No_of_labeled_points * 2
                Average_Polar_Radius_temp2(i,j) = 0;
                Congruence_row(i) = 0;
            end
        end
    end
end
end

%Eliminate discrete no_of_rows where consecutive congruence frames <
%(Min_Slow_Movement_Frames = 4)
Average_Polar_Radius_temp3 = zeros(no_of_rows, No_of_labeled_points*2);
for i=1:no_of_rows-1
    if(Congruence_row(i,1) < 1 && Congruence_row(i+1,1) >= 1)
        %transition to start of inspection fragment
        Start_fragment = i+1;
    elseif (Congruence_row(i,1) >= 1 && Congruence_row(i+1,1) < 1)
        %transition to end of inspection fragment
        End_fragment = i;
        if (End_fragment - Start_fragment + 1) >= Min_Slow_Movement_Frames
            for k=Start_fragment:End_fragment
                for j=1:No_of_labeled_points*2
                    Average_Polar_Radius_temp3(k,j) = Average_Polar_Radius_temp2(k,j);
                end
            end
        else
            for k=Start_fragment:End_fragment
                Congruence_row(k,1)=0;
            end
        end
    end
end

```

```

        end
    end
end
Average_Polar_Radius_temp2 = Average_Polar_Radius_temp3;

%Eliminate fragments where all Palms are moving in one direction
k=1;
palm_positive = 0;
palm_negative = 0;
Area_statistics = zeros(100,area_statistics_columns);
for i=1:no_of_rows-1
    if(Congruence_row(i,1) < 1 && Congruence_row(i+1,1) >= 1)
        %transition to start of inspection fragment
        Start_fragment = i+1;
    elseif (Congruence_row(i,1) >= 1 && Congruence_row(i+1,1) < 1)
        %transition to end of inspection fragment
        End_fragment = i;

        Area_statistics(k,1) = Start_fragment;
        Area_statistics(k,2) = End_fragment;

        if Average_Polar_Radius_temp2(i, No_of_labeled_points * 2) >= 0
            palm_positive = palm_positive + 1;
        else
            palm_negative = palm_negative + 1;
        end

        Area_statistics(k,3) = palm_positive;
        Area_statistics(k,4) = palm_negative;
        k = k+1;
    else
        if Average_Polar_Radius_temp2(i, No_of_labeled_points * 2) >= 0
            palm_positive = palm_positive + 1;
        else
            palm_negative = palm_negative + 1;
        end
    end
end
Area_statistics(~any(Area_statistics,2), : ) = [];

No_of_fragments = size(Area_statistics,1);
Average_Polar_Radius_temp3 = Average_Polar_Radius_temp2;
m = 1;
for k=1:No_of_fragments
    %if fragment either all moving forward or moving backward
    %eliminate fragment from wall rearing evaluation
    if(Area_statistics(k,3) == 0 || Area_statistics(k,4) == 0)
        Start_fragment = Area_statistics(k,1);
        End_fragment = Area_statistics(k,2);
        for i=Start_fragment:End_fragment
            for j=1:No_of_labeled_points * 2

```

```

        Average_Polar_Radius_temp3(i,j) = 0;
    end
    Congruence_row(i,1)=0;
end
else
    Area_statistics_temp(m,:) = Area_statistics(k,:);
    m = m + 1;
end
end
Area_statistics = Area_statistics_temp;
Average_Polar_Radius_temp2 = Average_Polar_Radius_temp3;
No_of_fragments = size(Area_statistics,1);

Average_Polar_Radius_temp3 = Average_Polar_Radius_temp2;
Fragment_Polar_Radius = zeros(no_of_rows,No_of_labeled_points * 3);
for k=1:No_of_fragments
    %Acquire start and end of fragment, and compute its duration
    Start_fragment = Area_statistics(k,1);
    End_fragment = Area_statistics(k,2);
    Duration_of_fragment = End_fragment - Start_fragment +1;

    for i=Start_fragment:End_fragment
        for j=1:No_of_labeled_points
            %Compute differential radial distance between (FingerTips to
            %Palm) to Fragment Array
            Fragment_Polar_Radius(i,j+No_of_labeled_points) = Average_Polar_Radius_temp3(i, ✓
j+No_of_labeled_points);
            Fragment_Polar_Radius(i,j) = Average_Polar_Radius_temp3(i,j)-Average_Polar_Radius_temp3(i, ✓
No_of_labeled_points);
        end
    end
    for m=Start_fragment:End_fragment
        %Flag all fingertips where differential radial distance are less
        %than threshold
        Fragment_Polar_Radius(m,3*No_of_labeled_points) = 0;
        for j = 1:No_of_labeled_points-1
            if abs(Fragment_Polar_Radius(m, j+No_of_labeled_points))< Threshold && abs ✓
(Fragment_Polar_Radius(m, j+No_of_labeled_points)) > 0
                Fragment_Polar_Radius(m,j+2*No_of_labeled_points) = 1;
                Fragment_Polar_Radius(m,3*No_of_labeled_points) = Fragment_Polar_Radius(m, ✓
3*No_of_labeled_points) + 1;
            else
                Fragment_Polar_Radius(m,j+2*No_of_labeled_points) = 0;
            end
        end
    end
end
end
end

```

```

%***** Extract Coherent Subfragment Specifications *****

%Initialize wall rearing fragment table
Fragment_data_warehouse = zeros(1,Columns_of_SubFragments);
Fragment_data_aux = zeros(1,17);

for k=1:No_of_fragments
    %1) For each fragment, further decompose into SubFragments based on three
    %most stable fingertips moving less than Min_palm_threshold;
    %
    %2) Followed by consolidation of SubFragments where consecutive
    %subfragments are only separated by 1 frame, to avoid unnecessary
    %subfragmentations due to minor obstructions;
    %
    %3) Compute statistics +/- 0.5 seconds around the consolidated
    %subfragments, to determine what the particular paw did prior to its
    %slow movements. Hence discern of wall rearing / floor rearing
    %activities

    Start_fragment = Area_statistics(k,1);
    End_fragment = Area_statistics(k,2);
    Duration_of_fragment = End_fragment - Start_fragment +1;

    %***** Step 1 *****
    %1) Identify the top three fingertips that demonstrate stability, and
    %    record its start and end sub_fragment timings;

    %2) Identify the stable Palm position within the sub_fragment timing
    %    where consecutive radial distance are less than Max_Palm_Threshold;
    %
    %3) Compute the average radial distance of the finger tips and the palm;
    %

    %Step 1a): Identify Stable Fingertips
    paw = zeros(No_of_labeled_points-1,2);
    paw(1,2) = 1; % 1 = Thumb finger
    paw(2,2) = 2; % 2 = index finger
    paw(3,2) = 3; % 3 = Middle finger
    paw(4,2) = 4; % 4 = Ring finger
    paw(5,2) = 5; % 5 = small finger

    Sum_of_minFingerPalm = 0;
    Sum_of_maxFingerPalm = 0;
    Average_stable_1st = 0;
    Average_stable_2nd = 0;
    Average_stable_3rd = 0;
    Average_palm = 0;

    for i=Start_fragment:End_fragment

```

```

%Compute for individual finger how many frames demonstrate slow
%movements ie ?(radial distance) < Max_Palm_Threshold
for j = 2*No_of_labeled_points+1:2*No_of_labeled_points+5
    if Fragment_Polar_Radius(i,j) > 0
        switch j
            case 2*No_of_labeled_points+1
                paw(1,1) = paw(1,1) + 1;
            case 2*No_of_labeled_points+2
                paw(2,1) = paw(2,1) + 1;
            case 2*No_of_labeled_points+3
                paw(3,1) = paw(3,1) + 1;
            case 2*No_of_labeled_points+4
                paw(4,1) = paw(4,1) + 1;
            otherwise
                paw(5,1) = paw(5,1) + 1;
        end
    end
end
%Identify the top 3 fingertips that are most stable within the
%fragment
sort_paw = sortrows(paw, 'descend');
stable_1st = int8(sort_paw(1,2));
stable_2nd = int8(sort_paw(2,2));
stable_3rd = int8(sort_paw(3,2));

Average_stable_1st = Average_stable_1st + Average_Polar_Radius_temp3(i,stable_1st);
Average_stable_2nd = Average_stable_2nd + Average_Polar_Radius_temp3(i,stable_2nd);
Average_stable_3rd = Average_stable_3rd + Average_Polar_Radius_temp3(i,stable_3rd);
Palm_rad_ii(i-Start_fragment+1)=Average_Polar_Radius_temp3(i,6);
Average_palm = Average_palm + Average_Polar_Radius_temp3(i,6);
end

Area_statistics(k,4+stable_1st) = Average_stable_1st / Duration_of_fragment;
Area_statistics(k,4+stable_2nd) = Average_stable_2nd / Duration_of_fragment;
Area_statistics(k,4+stable_3rd) = Average_stable_3rd / Duration_of_fragment;
Area_statistics(k,10) = Average_palm / Duration_of_fragment;

%Step 1b) Parse Stable FingerTips information to working array
Fragment_Polar_Radius_tmp = zeros(Duration_of_fragment, 6*No_of_labeled_points);
for i=Start_fragment:End_fragment
    for j = 1:No_of_labeled_points
        switch j
            case stable_1st
                Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
            case stable_2nd
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
            case stable_3rd
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
            otherwise
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j) = ✓
        end
    end
end

```



```

Fragment_Polar_Radius(i,j)-Fragment_Polar_Radius(i-1,j);
    if abs(Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j)) < ✓
Threshold
        %(FingerTip - Palm distance moves less than 5
        %pixels per frame
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 1;
    else
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 0;
    end
    case stable_2nd
        Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,No_of_labeled_points+j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,2*No_of_labeled_points+j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j)-Fragment_Polar_Radius(i-1,j);
    if abs(Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j)) < ✓
Threshold
        %(FingerTip - Palm distance moves less than 5
        %pixels per frame
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 1;
    else
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 0;
    end
    case stable_3rd
        Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,No_of_labeled_points+j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,2*No_of_labeled_points+j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j)-Fragment_Polar_Radius(i-1,j);
    if abs(Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j)) < ✓
Threshold
        %(FingerTip - Palm distance moves less than 5
        %pixels per frame
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 1;
    else
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 0;
    end
    case No_of_labeled_points
        Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,No_of_labeled_points+j);
    end

```



```

end
%Tag frames that exhibit coherent slow movement in FingerTips -
%Palm distances
Total_slow_mvt_pt = 0;
for j=1:5
    Total_slow_mvt_pt = Total_slow_mvt_pt + Fragment_Polar_Radius_tmp(i-Start_fragment+1, ✓
5*No_of_labeled_points+j);
end
if Total_slow_mvt_pt >= Min_Congruence
    Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points) = 1;
else
    Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points) = 0;
end
end

%
%During wall-rearing, fingertips will demonstrate temporal stability,
%that FingerTips - Palm shall exhibit short period of small changes
%between frames < Minimum Movement Threshold
%
%2a)Gauge sub-fragments within the fragment where palm exhibit
% small changes in (FingerTips - Palm distance);
%
%2b)Consolidate the two sub-fragments into one sub-fragment array before
% proceeding to Step 3)

%----- Step 2a -----
%Step 2a) Gauge subfragments that show coherent minimal changes in
% (FingerTips - Palm) distance

No_of_SubFragments = 100;
SubFrag_counter = 1;
SubFragment_stat_prior = zeros(No_of_SubFragments, 3*No_of_labeled_points+4);

Start_SubFragment = 1;
End_SubFragment = 1;
SubFragment_stat = zeros(No_of_SubFragments,Columns_of_SubFragments);

for m=1:Duration_of_fragment
    if m <= Duration_of_fragment-1
        %Within fragment treatment
        if Fragment_Polar_Radius_tmp(m,5*No_of_labeled_points) < 1 && Fragment_Polar_Radius_tmp ✓
(m+1,5*No_of_labeled_points) == 1
            Start_SubFragment = m+1;
        elseif Fragment_Polar_Radius_tmp(m,5*No_of_labeled_points) == 1 && ✓
Fragment_Polar_Radius_tmp(m+1,5*No_of_labeled_points) < 1
            End_SubFragment = m;
            SubFragment_stat_prior(SubFrag_counter,1) = round((Start_SubFragment + End_SubFragment) ✓
/2);
            SubFragment_stat_prior(SubFrag_counter,2) = Start_SubFragment;
            SubFragment_stat_prior(SubFrag_counter,3) = End_SubFragment;

```

```

        SubFrag_counter = SubFrag_counter + 1;
        Start_SubFragment = 1;
        End_SubFragment = 1;
    end
else
    %End of fragment treatment where the last frame has coherence
    if Fragment_Polar_Radius_tmp(m,5*No_of_labeled_points) == 1
        End_SubFragment = Duration_of_fragment;
        SubFragment_stat_prior(SubFrag_counter,1) = round((Start_SubFragment + End_SubFragment) / 2);
        SubFragment_stat_prior(SubFrag_counter,2) = Start_SubFragment;
        SubFragment_stat_prior(SubFrag_counter,3) = End_SubFragment;
    end

    SubFrag_counter = 1;
    Start_SubFragment = 1;
    End_SubFragment = 1;
end

%Compact SubFragment Array
SubFragment_stat_prior(~any(SubFragment_stat_prior,2), :) = [];
SubFragment_stat_prior_tmp = zeros(1,3*No_of_labeled_points+4);
n=1;
for m=1:size(SubFragment_stat_prior,1)
    %remove subfragment that has Start and End Subfragment on same row
    if SubFragment_stat_prior(m,2) < SubFragment_stat_prior(m,3)
        SubFragment_stat_prior_tmp(n,:) = SubFragment_stat_prior(m,:);
        n=n+1;
    end
end
SubFragment_stat_prior = SubFragment_stat_prior_tmp;

%Consolidate SubFragments, such that subfragments separated less than
%the Max_SubFragment_Gap are considered as one subfragment
SubFragment_gap = zeros(100,1);
for m=1:99
    if ( m < size(SubFragment_stat_prior,1))
        SubFragment_gap(m) = SubFragment_stat_prior(m+1,2) - SubFragment_stat_prior(m,3);
        if SubFragment_gap(m) < Max_SubFragment_Gap
            SubFragment_stat_prior(m,3) = SubFragment_stat_prior(m+1,3);
            SubFragment_stat_prior(m+1, :) = [];
        end
    end
end

SubFragment_stat_latter = SubFragment_stat_prior;
for m=1:size(SubFragment_stat_prior,1)
    SubFragment_stat(m,1) = SubFragment_stat_prior(m,1)+Start_fragment;
    SubFragment_stat(m,2) = SubFragment_stat_prior(m,2)+Start_fragment;
    SubFragment_stat(m,3) = SubFragment_stat_prior(m,3)+Start_fragment;
end
SubFragment_stat(~any(SubFragment_stat,2), :) = [];

```

```

No_of_SubFragments = size(SubFragment_stat,1);
SubFragment_aux = zeros(No_of_SubFragments,17);

%Step 3)For each sub-fragment, average change in fingertips shall be
% smaller than average change in palm coordinates during
% wall-rearing; as fingertips movement are constrained while palm are
% free
%
for p=1:No_of_SubFragments

    Start_SubFragment = SubFragment_stat_prior(p,2)+Start_fragment;
    End_SubFragment = SubFragment_stat_prior(p,3)+Start_fragment;

    Start_prior = Start_SubFragment - Inspection_window;
    End_latter = End_SubFragment + Inspection_window;
    Duration_inspection_window = End_latter - Start_prior + 1;
    if Start_prior < 1
        Start_prior = 1;
    end
    if End_latter > no_of_rows
        End_latter = no_of_rows;
    end

    %Compute Statistics prior to congruence point
    Sum_stable_1st_radius = 0;
    Sum_stable_2nd_radius = 0;
    Sum_stable_3rd_radius = 0;
    Sum_stable_palm = 0;

    Sum_stable_1st_radius_delta = 0;
    Sum_stable_2nd_radius_delta = 0;
    Sum_stable_3rd_radius_delta = 0;
    Sum_stable_palm_delta = 0;

    for q = Start_prior:Start_SubFragment
        %Compute average radial distance prior to Mid-subfragment
        Sum_stable_1st_radius = Sum_stable_1st_radius + Average_Polar_Radius(q,stable_1st);
        Sum_stable_2nd_radius = Sum_stable_2nd_radius + Average_Polar_Radius(q,stable_2nd);
        Sum_stable_3rd_radius = Sum_stable_3rd_radius + Average_Polar_Radius(q,stable_3rd);
        Sum_stable_palm = Sum_stable_palm + Average_Polar_Radius(q,No_of_labeled_points);

        %Compute total radial distance change prior to Start-SubFragment
        Sum_stable_1st_radius_delta = Sum_stable_1st_radius_delta + Average_Polar_Radius(q, ✓
stable_1st+No_of_labeled_points);
        Sum_stable_2nd_radius_delta = Sum_stable_2nd_radius_delta + Average_Polar_Radius(q, ✓
stable_2nd+No_of_labeled_points);
        Sum_stable_3rd_radius_delta = Sum_stable_3rd_radius_delta + Average_Polar_Radius(q, ✓
stable_3rd+No_of_labeled_points);
        Sum_stable_palm_delta = Sum_stable_palm_delta + Average_Polar_Radius(q, ✓
2*No_of_labeled_points);
    end
end

```

```

end

Duration_of_SubFragment = Start_SubFragment - Start_prior;
if (Duration_of_SubFragment > 0)
    %Average Radial distances of FingerTips & Palm
    SubFragment_stat_prior(p, 3+stable_1st) = Sum_stable_1st_radius / double ✓
(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+stable_2nd) = Sum_stable_2nd_radius / double ✓
(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+stable_3rd) = Sum_stable_3rd_radius / double ✓
(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+No_of_labeled_points) = Sum_stable_palm / double ✓
(Duration_of_SubFragment);
    %Average change in FingerTips & Palm per frame
    SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_1st) = ✓
Sum_stable_1st_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_2nd) = ✓
Sum_stable_2nd_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_3rd) = ✓
Sum_stable_3rd_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+3*No_of_labeled_points) = Sum_stable_palm_delta / double ✓
(Duration_of_SubFragment);
else
    SubFragment_stat_prior(p,:) = 0;
end

%Compute Statistics after congruence point
Sum_stable_1st_radius = 0;
Sum_stable_2nd_radius = 0;
Sum_stable_3rd_radius = 0;
Sum_stable_palm = 0;

Sum_stable_1st_radius_delta = 0;
Sum_stable_2nd_radius_delta = 0;
Sum_stable_3rd_radius_delta = 0;
Sum_stable_palm_delta = 0;

for q = End_SubFragment:End_latter
    %Compute average radial distance prior to Mid-subfragment
    Sum_stable_1st_radius = Sum_stable_1st_radius + Average_Polar_Radius(q,stable_1st);
    Sum_stable_2nd_radius = Sum_stable_2nd_radius + Average_Polar_Radius(q,stable_2nd);
    Sum_stable_3rd_radius = Sum_stable_3rd_radius + Average_Polar_Radius(q,stable_3rd);
    Sum_stable_palm = Sum_stable_palm + Average_Polar_Radius(q,No_of_labeled_points);

    %Compute total radial distance change prior to Mid-SubFragment
    Sum_stable_1st_radius_delta = Sum_stable_1st_radius_delta + Average_Polar_Radius(q, ✓
stable_1st+No_of_labeled_points);
    Sum_stable_2nd_radius_delta = Sum_stable_2nd_radius_delta + Average_Polar_Radius(q, ✓
stable_2nd+No_of_labeled_points);
    Sum_stable_3rd_radius_delta = Sum_stable_3rd_radius_delta + Average_Polar_Radius(q, ✓
stable_3rd+No_of_labeled_points);
    Sum_stable_palm_delta = Sum_stable_palm_delta + Average_Polar_Radius(q, ✓

```



```

2*No_of_labeled_points);

end

Duration_of_SubFragment = End_latter - End_SubFragment;
if (Duration_of_SubFragment > 0)
    %Average Radial distances of FingerTips & Palm
    SubFragment_stat_latter(p, 3+stable_1st) = Sum_stable_1st_radius / double ✓
(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+stable_2nd) = Sum_stable_2nd_radius / double ✓
(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+stable_3rd) = Sum_stable_3rd_radius / double ✓
(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+No_of_labeled_points) = Sum_stable_palm / double ✓
(Duration_of_SubFragment);
    %Average change in FingerTips & Palm per frame
    SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_1st) = ✓
Sum_stable_1st_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_2nd) = ✓
Sum_stable_2nd_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_3rd) = ✓
Sum_stable_3rd_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+3*No_of_labeled_points) = Sum_stable_palm_delta / double ✓
(Duration_of_SubFragment);
else
    SubFragment_stat_latter(p,:) = 0;
end

%**** Compute statistics for entire SubFragment ****%
Radii_1 = zeros(End_latter - Start_prior + 1,1);
Radii_2 = zeros(End_latter - Start_prior + 1,1);
Radii_3 = zeros(End_latter - Start_prior + 1,1);
Palm_1 = zeros(End_latter - Start_prior + 1,1);

for q = Start_prior:End_latter
    %Compute average radial distance prior to Mid-subfragment
    Radii_1(q - Start_prior + 1) = Average_Polar_Radius(q,stable_1st);
    Radii_2(q - Start_prior + 1) = Average_Polar_Radius(q,stable_2nd);
    Radii_3(q - Start_prior + 1) = Average_Polar_Radius(q,stable_3rd);
    Palm_1(q - Start_prior + 1) = Average_Polar_Radius(q,No_of_labeled_points);
end

SubFragment_aux(p,1) = Start_SubFragment;
SubFragment_aux(p,2) = End_SubFragment;
SubFragment_aux(p,2+stable_1st) = mean(Radii_1);
SubFragment_aux(p,2+stable_2nd) = mean(Radii_2);
SubFragment_aux(p,2+stable_3rd) = mean(Radii_3);
SubFragment_aux(p,2+No_of_labeled_points) = mean(Palm_1);
SubFragment_aux(p,10+stable_1st) = std(Radii_1);
SubFragment_aux(p,10+stable_2nd) = std(Radii_2);
SubFragment_aux(p,10+stable_3rd) = std(Radii_3);
SubFragment_aux(p,10+No_of_labeled_points) = std(Palm_1);

```

```

outward = 0;
if SubFragment_aux(p,2+stable_1st) > SubFragment_aux(p,2+No_of_labeled_points)
    outward = outward + 1;
end
if SubFragment_aux(p,2+stable_2nd) > SubFragment_aux(p,2+No_of_labeled_points)
    outward = outward + 1;
end
if SubFragment_aux(p,2+stable_3rd) > SubFragment_aux(p,2+No_of_labeled_points)
    outward = outward + 1;
end
%Compute Fingertips in outward orientation
SubFragment_aux(p,3+No_of_labeled_points)=outward;

below_inner = 0;
if SubFragment_aux(p,2+stable_1st) <= Inner_Diameter * safety_margin / 2
    below_inner = below_inner + 1;
end
if SubFragment_aux(p,2+stable_2nd) <= Inner_Diameter * safety_margin / 2
    below_inner = below_inner + 1;
end
if SubFragment_aux(p,2+stable_3rd) <= Inner_Diameter * safety_margin / 2
    below_inner = below_inner + 1;
end
%Compute Fingertips within inner diameter
SubFragment_aux(p,4+No_of_labeled_points)=below_inner;

SubFragment_aux(p,17) = (SubFragment_aux(p,2+No_of_labeled_points)+SubFragment_aux(p, 2+stable_1st)+SubFragment_aux(p,2+stable_2nd)+SubFragment_aux(p,2+stable_3rd))/4;

%Discern fingertips and palm movement dynamics within
%congruent subfragment ie +/- 0.5 seconds from coherent minima
stable_1st_mvt = mvtSymbol(SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_1st),
SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_1st));
stable_2nd_mvt = mvtSymbol(SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_2nd),
SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_2nd));
stable_3rd_mvt = mvtSymbol(SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_3rd),
SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_3rd));
stable_palm_mvt = mvtSymbol(SubFragment_stat_prior(p, 3+3*No_of_labeled_points),
SubFragment_stat_latter(p, 3+3*No_of_labeled_points));

SubFragment_stat(p,2)=Start_SubFragment;
SubFragment_stat(p,3+stable_1st) = stable_1st_mvt;
SubFragment_stat(p,3+stable_2nd) =stable_2nd_mvt;
SubFragment_stat(p,3+stable_3rd) =stable_3rd_mvt;
SubFragment_stat(p,9) =stable_palm_mvt;

SubFragment_stat(p,10) = touchDetect(Start_SubFragment, End_SubFragment, stable_1st_mvt,
stable_2nd_mvt, stable_3rd_mvt, stable_palm_mvt);

end

```



```

Fragment_header = zeros(1,Columns_of_SubFragments);
Fragment_header(1,1) = Start_fragment;
Fragment_header(1,2) = End_fragment;
Fragment_data_warehouse = cat(1,Fragment_data_warehouse,Fragment_header);
Fragment_data_warehouse = cat(1,Fragment_data_warehouse,SubFragment_stat);

Fragment_aux_header = zeros(1,17);
Fragment_aux_header(1,1) = Start_fragment;
Fragment_aux_header(1,2) = End_fragment;
Fragment_data_aux = cat(1,Fragment_data_aux,Fragment_aux_header);
Fragment_data_aux = cat(1,Fragment_data_aux,SubFragment_aux);

end
Fragment_data_warehouse(1,:) = [] ;
Fragment_data_aux(1,:) = [] ;

%***** Extract Congruent Points Specifications *****
%Initialize wall rearing fragment table
Fragment_congruence = zeros(1,10);

for k=1:No_of_fragments

    %For each fragment, inspect existence of the following evidence to
    %confirm of wall rearing
    %
    %Three possible situations will happen as paw hits the wall, be obstructed
    %and bounced back
    %
    %1a)Distance between finger tips and palm
    % shall demonstrate congruent minima for some fingers while decreasing
    % distance for others; it will not be surprising, due to acrylic
    % reflection, that some of these distances are negative, as DLC
    % recognize mirror reflection of palm based on its virtual image.
    %
    %1b)Change in radial distances of fingertips and palm from the center
    % shall be peaked (and possibly reversed)
    %
    %1c)Fingertips may linger on the wall (ie change of fingertips radial
    % distance will be hovering < 1 pics per frame

    %Acquire start and end of fragment, and compute its duration

    Start_fragment = Area_statistics(k,1);
    End_fragment = Area_statistics(k,2);
    Duration_of_fragment = End_fragment - Start_fragment +1;

    %
    %1) Identify the top three fingertips that demonstrate stability, and
    % record its start and end sub_fragment timings;

    %2) Identify the stable Palm position within the sub_fragment timing
    % where consecutive radial distance are less than Max_Palm_Threshold;

```

```

%
%3) Compute the average radial distance of the finger tips and the palm;
%
%4) Check if Palm radial distance is less than finger tips; if yes, paw
%   is oriented outwards, else it is oriented inwards

paw = zeros(No_of_labeled_points,2);
paw(1,2) = 1; % 1 = Thumb finger
paw(2,2) = 2; % 2 = index finger
paw(3,2) = 3; % 3 = Middle finger
paw(4,2) = 4; % 4 = Ring finger
paw(5,2) = 5; % 5 = small finger
paw(6,2) = 6; % 6 = palm
for i=Start_fragment:End_fragment
    %Compute for individual finger how many frames demonstrate slow
    %movements ie ?(radial distance) < Max_Palm_Threshold
    for j = 2*No_of_labeled_points+1:2*No_of_labeled_points+5
        if Fragment_Polar_Radius(i,j) > 0
            switch j
                case 2*No_of_labeled_points+1
                    paw(1,1) = paw(1,1) + 1;
                case 2*No_of_labeled_points+2
                    paw(2,1) = paw(2,1) + 1;
                case 2*No_of_labeled_points+3
                    paw(3,1) = paw(3,1) + 1;
                case 2*No_of_labeled_points+4
                    paw(4,1) = paw(4,1) + 1;
                otherwise
                    paw(5,1) = paw(5,1) + 1;
            end
        end
    end
    %Identify the top 3 fingertips that are most stable within the
    %fragment
    sort_paw = sortrows(paw, 'descend');
    stable_1st = int8(sort_paw(1,2));
    stable_2nd = int8(sort_paw(2,2));
    stable_3rd = int8(sort_paw(3,2));
end

Fragment_Polar_Radius_tmp = zeros(Duration_of_fragment, No_of_labeled_points * 4);
for i=Start_fragment:End_fragment
    for j = 1:No_of_labeled_points
        switch j
            case stable_1st
                Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
            case stable_2nd
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
            case stable_3rd
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
        end
    end
end
Fragment_Polar_Radius(i,2*No_of_labeled_points+j);

```

```

        case stable_2nd
            Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
        Fragment_Polar_Radius(i,j);
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
        Fragment_Polar_Radius(i,No_of_labeled_points+j);
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
        Fragment_Polar_Radius(i,2*No_of_labeled_points+j);
        case stable_3rd
            Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
        Fragment_Polar_Radius(i,j);
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
        Fragment_Polar_Radius(i,No_of_labeled_points+j);
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
        Fragment_Polar_Radius(i,2*No_of_labeled_points+j);
        case No_of_labeled_points
            Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
        Fragment_Polar_Radius(i,j);
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
        Fragment_Polar_Radius(i,No_of_labeled_points+j);
            %Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
        Fragment_Polar_Radius(i,2*No_of_labeled_points+j);
    end
end
end

%1) Identify sub-fragments within the fragment where palm is shortest
% distance (ie Congruent Minima) from the stable fingertips; in these
% settings the paw is either grabbing something
% or pushing against the wall
%
%2) Back-trace the minima to the last moving palm to establish
% sub-fragment
%
%3) For each sub-fragment, average change in fingertips shall be
% smaller than average change in palm coordinates during
% wall-rearing; as fingertips movement are constrained while palm are
% free
%
for m=1:Duration_of_fragment
    %Flag stable palm sub-fragment
    if abs(Fragment_Polar_Radius_tmp(m,No_of_labeled_points*3)) <= Max_Palm_Threshold
        % ?(Palm) < Max_Palm_Threshold => Flag Stable Palm
        Fragment_Polar_Radius_tmp(m,No_of_labeled_points*4) = 1;
    else
        Fragment_Polar_Radius_tmp(m,No_of_labeled_points*4) = 0;
    end
end
end
%Extract minima fingertips - Palm distances
Stable_1st_finger = zeros(Duration_of_fragment, 1);

```

```

Stable_2nd_finger = zeros(Duration_of_fragment, 1);
Stable_3rd_finger = zeros(Duration_of_fragment, 1);
for m=1:Duration_of_fragment
    Stable_1st_finger(m,1) = -Fragment_Polar_Radius_tmp(m,No_of_labeled_points+stable_1st);
    Stable_2nd_finger(m,1) = -Fragment_Polar_Radius_tmp(m,No_of_labeled_points+stable_2nd);
    Stable_3rd_finger(m,1) = -Fragment_Polar_Radius_tmp(m,No_of_labeled_points+stable_3rd);

    [Min_1st, Min_1st_idx] = findpeaks(Stable_1st_finger);
    [Min_2nd, Min_2nd_idx] = findpeaks(Stable_2nd_finger);
    [Min_3rd, Min_3rd_idx] = findpeaks(Stable_3rd_finger);
end

%Group the trough points into clusters into sub_fragments
Congruent_Minima_index = zeros(1000,3);
Congruent_Mode_index = zeros(1000,1);
for l=1:size(Min_1st_idx)
    Congruent_Minima_index(int16(Min_1st_idx(1)),1)=1;
end
for l=1:size(Min_2nd_idx)
    Congruent_Minima_index(int16(Min_2nd_idx(1)),2)=1;
end
for l=1:size(Min_3rd_idx)
    Congruent_Minima_index(int16(Min_3rd_idx(1)),3)=1;
end
for p=1:1000-Congruence_window
    %where minima is observed, inspect next neighbor frames where
    %minima will also be observed
    if Congruent_Minima_index(p,1) > 0 || Congruent_Minima_index(p,2) > 0 || Congruent_Minima_index
    (p,3) > 0
        for q=0:Congruence_window
            Congruent_Mode_index (p) = Congruent_Mode_index (p) + Congruent_Minima_index(p+q,1)+
Congruent_Minima_index (p+q,2) + Congruent_Minima_index(p+q,3);
        end
    end
end
%Find congruence points where minima of fingertips - palm distances
%occur within 1 second among each other
[Congr_peaks_tmp, Congr_peaks_loc_tmp] = findpeaks(Congruent_Mode_index, 'MinPeakDistance',
Congruence_window);

r = 1;
Congr_peaks_loc = zeros(100,1);
Congr_peaks_loc_tmp1 = Congr_peaks_loc_tmp;
for p=1:size(Congr_peaks_loc_tmp)
    if Congr_peaks_tmp(p) >= Min_Congruence
        %Fine tune location of Congruence Peaks
        %By extracting median within the Congruence_window
        Start_Congruence = int16(Congr_peaks_loc_tmp(p));
        Congr_median_array = zeros(3*(Congruence_window),1);
        for u=0:Congruence_window - 1
            for v = 1:3
                if Congruent_Minima_index(Start_Congruence + u, v) > 0

```



```

        Congr_median_array(3*u+v) = Start_Congruence + u;
    end
end
end
Congr_median_array = Congr_median_array(Congr_median_array ~= 0);
Congr_peaks_loc_tmpl(p) = median(Congr_median_array);

Congr_peaks_loc(r) = Congr_peaks_loc_tmpl(p);
r=r+1;
end
end
Congr_peaks_loc = Congr_peaks_loc(Congr_peaks_loc ~= 0);

No_of_SubFragments = size(Congr_peaks_loc,1);
SubFragment_stat_prior = zeros(No_of_SubFragments, 3*No_of_labeled_points+1);
SubFragment_stat_latter = zeros(No_of_SubFragments, 3*No_of_labeled_points+1);
Start_SubFragment = 1;
Mid_SubFragment = 1;
End_SubFragment = 1;
SubFragment_stat = zeros(No_of_SubFragments,10);
for p=1:No_of_SubFragments
    % Locate mid-points of maximum congruence between 3 stable
    % fingertips
    Mid_SubFragment = int16(Congr_peaks_loc(p))+Start_fragment;

    if Mid_SubFragment - Inspection_window > 1
        Start_SubFragment = Mid_SubFragment - Inspection_window;
    else
        Start_SubFragment = 1;
    end
    if Mid_SubFragment + Inspection_window < no_of_rows
        End_SubFragment = Mid_SubFragment + 12;
    else
        End_SubFragment = no_of_rows;
    end

    %*****Save Congruence Points *****

    SubFragment_stat(p,1) = Mid_SubFragment;
    SubFragment_stat(p,2) = Start_SubFragment;
    SubFragment_stat(p,3) = End_SubFragment;

end

Fragment_congruence = cat(1,Fragment_congruence,SubFragment_stat);

end
Fragment_congruence(1,:) = [] ;

%*****Decisions Evaluation Specifications *****
%**** 1st Phase: Correlate Coherent Subfragments with Congruence Points

```

```

%**** initial gauge of wall rearing

No_coherent_Subfragments = size(Fragment_data_warehouse,1);
No_congruent_points = size(Fragment_congruence,1);

%Fragment_data_warehouse should only flag subfragments that are BOTH
%Coherent (i.e. three fingers moving below 5 pixels per frame) AND
%Congruence within 1 second of Coherence (i.e. FingerTips - Palm distance
%of slow moving fingers exhibi minima)

Fragment_data_warehouse_tmp = zeros(No_coherent_Subfragments, Columns_of_SubFragments);
for i=1:No_coherent_Subfragments
    if Fragment_data_warehouse(i,3) > 0 && Fragment_data_warehouse(i,10) > 0
        %Process Coherent SubFragment records which was tagged as plausible
        %wall-rearing
        %
        %For each subfragment
        %if Coherent Subfragment +/- 1 second possess congruence points
        %decision(s) = 1;
        %else decision = 0
        %
        Fragment_data_warehouse_tmp(i,:)=Fragment_data_warehouse(i,:);
        Start_SubFragment = Fragment_data_warehouse(i,2)-Frame_Rate;
        End_SubFragment = Fragment_data_warehouse(i,3)+Frame_Rate;
        No_of_congruence = 0;
        for j=1:No_congruent_points
            %Find Number of Congruence points within each coherent
            %subfragment
            if Fragment_congruence(j,1)>=Start_SubFragment && Fragment_congruence(j,1) ✓
                <=End_SubFragment
                    No_of_congruence = No_of_congruence + 1;
                end
            end
            if No_of_congruence > 0
                Fragment_data_warehouse_tmp(i,10) = 1;
                Fragment_data_warehouse_tmp(i,11) = No_of_congruence;
            else
                Fragment_data_warehouse_tmp(i,10) = 0;
                Fragment_data_warehouse_tmp(i,11) = 0;
            end
        end
        %save header records
        Fragment_data_warehouse_tmp(i,:) = Fragment_data_warehouse(i,:);
    end
end

Fragment_data_warehouse = Fragment_data_warehouse_tmp;

%**** 2nd Phase: Eliminate Stationary Movements
%***** Filter Subfragments that are unlikely to be wall rearing
%1) If Subfragment's palm demonstrate standard deviations <

```



```

% Crawling_low_threshold
%
%2) If Subfragment duration is less than Minimum_touch_duration
%
Fragment_data_warehouse_tmp = zeros(No_coherent_Subfragments, Columns_of_SubFragments);
for i=1:No_coherent_Subfragments
    if Fragment_data_warehouse(i,3) > 0
        if Fragment_data_warehouse(i,10) > 0
            %transfer details records where wall-rearing is suspected
            Fragment_data_warehouse_tmp(i,:) = Fragment_data_warehouse(i,:);

            stationary_count = 0;
            for j=1:6
                if Fragment_data_aux(i,10+j) > 0
                    if Fragment_data_aux(i,10+j) <= Crawling_low_threshold
                        stationary_count = stationary_count + 1;
                    end
                end
            end
            if stationary_count > 1
                Fragment_data_warehouse_tmp(i,10)=0;
                Fragment_data_warehouse_tmp(i,11)=0;
            end

            if (Fragment_data_warehouse(i,3) - Fragment_data_warehouse(i,2)) < Minimum_touch_duration
                Fragment_data_warehouse_tmp(i,10)=0;
                Fragment_data_warehouse_tmp(i,11)=0;
            end

        else
            %save details record with no wall-rearing detected
            Fragment_data_warehouse_tmp(i,:) = Fragment_data_warehouse(i,:);
        end
    else
        Fragment_ref = Fragment_ref + 1;
        %save header records
        Fragment_data_warehouse_tmp(i,:) = Fragment_data_warehouse(i,:);
    end
end

%*** 3rd Phase: Evaluate Subfragment Statistics
%**** Program starts *****
Average_Polar_Radius_temp4 = zeros(no_of_rows, 12);
Average_Polar_Radius_temp5 = zeros(no_of_rows, 12);
Fragment_data_warehouse_tmp2 = zeros(1000,4);
Fragment_data_warehouse_tmp3 = zeros(1000,4);

Fragment_data_warehouse_tmp1 = Fragment_data_warehouse_tmp;
for i=1:No_coherent_Subfragments
    if Fragment_data_warehouse_tmp1(i,3) > 0
        if Fragment_data_warehouse_tmp1(i,10) > 0
            %Enlarge surveillance to +/- Inspection Windows of SubFragment

```

```

Start_SubFragment = int16(Fragment_data_warehouse_tmp1(i,2) - Inspection_window);
End_SubFragment = int16(Fragment_data_warehouse_tmp1(i,3) + Inspection_window);
if Start_SubFragment < 1
    Start_SubFragment = 1;
end
if End_SubFragment >= no_of_rows
    End_SubFragment = no_of_rows - 1;
end

%Investigate on enlarged subfragment
for k=Start_SubFragment:End_SubFragment
    for j = 1:6
        Average_Polar_Radius_temp4(k,j) = Raw_Polar_Radius(k,j);
        Average_Polar_Radius_temp4(k,j+No_of_labeled_points) = Raw_Polar_Radius(k+1,j)- ✓
Raw_Polar_Radius(k,j);
    end
end
end
end
end

Average_Polar_Radius_temp5 = Average_Polar_Radius_temp4;
for i=1:no_of_rows
    Transition_counter = 0;
    for j=No_of_labeled_points+1:2*No_of_labeled_points
        if abs(Average_Polar_Radius_temp4(i,j)) >= Transition_TH
            Transition_counter = Transition_counter + 1;
        end
    end
    if Transition_counter > 1
        %Number of fingertips that exceed Transition_TH >= 2
        for j=1:2*No_of_labeled_points
            Average_Polar_Radius_temp5(i,j) = 0;
        end
    end
end
end

%Repartition SubFragment
k=1;
Start_Subfragment = 1;
SubFragment_statistics_aux = zeros(1000,3+2*No_of_labeled_points);
for i=1:no_of_rows-1
    if(Average_Polar_Radius_temp5(i,1) < 1 && Average_Polar_Radius_temp5(i+1,1)>= 1)
        %transition to start of repartitioned Subfragment
        Start_Subfragment = i+1;
    elseif (Average_Polar_Radius_temp5(i,1) >= 1 && Average_Polar_Radius_temp5(i+1,1)< 1)
        %transition to end of repartitioned Subfragment
        End_Subfragment = i;

        SubFragment_statistics_aux(k,1) = Start_Subfragment;
        SubFragment_statistics_aux(k,2) = End_Subfragment;
    end
end

```

```

    k = k + 1;
elseif (Average_Polar_Radius_temp5(no_of_rows,1) >= 1 && i == no_of_rows-1)
    %End of array treatment
    End_Subfragment = no_of_rows;

    SubFragment_statistics_aux(k,1) = Start_Subfragment;
    SubFragment_statistics_aux(k,2) = End_Subfragment;
end
end
SubFragment_statistics_aux(~any(SubFragment_statistics_aux,2), : ) = [];

%Eliminate any repartitioned SubFragment that has less than 2 frames
No_repart_subfrag = size(SubFragment_statistics_aux,1);
for k=1:No_repart_subfrag
    if (SubFragment_statistics_aux(k,2) - SubFragment_statistics_aux(k,1) < Min_stat_computation)
        %If duration of frame transitions less than 3 time frames,
        %eliminate subfragments from statistics computation
        Start_SubFragment = int16(SubFragment_statistics_aux(k,1));
        End_SubFragment = int16(SubFragment_statistics_aux(k,2));
        for i=Start_SubFragment:End_SubFragment
            for j=1:2*No_of_labeled_points
                Average_Polar_Radius_temp5(i,j) = 0;
            end
        end
        SubFragment_statistics_aux(k,1) = 0;
        SubFragment_statistics_aux(k,2) = 0;
    end
end
SubFragment_statistics_aux(~any(SubFragment_statistics_aux,2), : ) = [];

%Compute statistics for each repartitioned SubFragment
No_repart_subfrag = size(SubFragment_statistics_aux,1);
for k=1:No_repart_subfrag
    Start_SubFragment = int16(SubFragment_statistics_aux(k,1));
    End_SubFragment = int16(SubFragment_statistics_aux(k,2));
    SubFragment_duration = End_SubFragment - Start_SubFragment + 1;

    finger_1=zeros(SubFragment_duration,1);
    finger_2=zeros(SubFragment_duration,1);
    finger_3=zeros(SubFragment_duration,1);
    finger_4=zeros(SubFragment_duration,1);
    finger_5=zeros(SubFragment_duration,1);
    palm=zeros(SubFragment_duration,1);

    m=1;
    for i=Start_SubFragment:End_SubFragment
        finger_1(m) = Average_Polar_Radius_temp5(i,1);
        finger_2(m) = Average_Polar_Radius_temp5(i,2);
        finger_3(m) = Average_Polar_Radius_temp5(i,3);
        finger_4(m) = Average_Polar_Radius_temp5(i,4);
    end
end

```

```

    finger_5(m) = Average_Polar_Radius_temp5(i,5);
    palm(m) = Average_Polar_Radius_temp5(i,6);
    m=m+1;
end

SubFragment_statistics_aux (k,3) = mean(finger_1);
SubFragment_statistics_aux (k,4) = mean(finger_2);
SubFragment_statistics_aux (k,5) = mean(finger_3);
SubFragment_statistics_aux (k,6) = mean(finger_4);
SubFragment_statistics_aux (k,7) = mean(finger_5);
SubFragment_statistics_aux (k,8) = mean(palm);

SubFragment_statistics_aux (k,3+No_of_labeled_points) = std2(finger_1);
SubFragment_statistics_aux (k,4+No_of_labeled_points) = std2(finger_2);
SubFragment_statistics_aux (k,5+No_of_labeled_points) = std2(finger_3);
SubFragment_statistics_aux (k,6+No_of_labeled_points) = std2(finger_4);
SubFragment_statistics_aux (k,7+No_of_labeled_points) = std2(finger_5);
SubFragment_statistics_aux (k,8+No_of_labeled_points) = std2(palm);

end

DigitPeaks = zeros(No_repart_subfrag,16);
DigitPeaksFlag = zeros(No_repart_subfrag,1);
for k=1:8
    DigitPeaks(:,k) = SubFragment_statistics_aux (:,k);
end
for k=3:7 %Find maxima of mean digit radii for each digits

    if DigitPeaks(1,k) > DigitPeaks(2,k)
        DigitPeaks(1,k+6) = 1;
    else
        DigitPeaks(1,k+6) = 0;
    end
    if DigitPeaks(No_repart_subfrag,k) > DigitPeaks(No_repart_subfrag-1,k)
        DigitPeaks(No_repart_subfrag,k+6) = 1;
    else
        DigitPeaks(No_repart_subfrag,k+6) = 0;
    end

    for l=2:(No_repart_subfrag-1)
        if DigitPeaks(1,k) > DigitPeaks(l-1,k) && DigitPeaks(1,k) > DigitPeaks(l+1,k)
            DigitPeaks(1,k+6) = 1;
        else
            DigitPeaks(1,k+6) = 0;
        end
    end
end

end
DigitPeaks(:,14) = DigitPeaks(:,9)+DigitPeaks(:,10)+DigitPeaks(:,11)+DigitPeaks(:,12)+DigitPeaks(:,13);
for m=1:No_repart_subfrag
    %Compute Convolution Column of DigitPeaks
    if m > 1 && m < No_repart_subfrag
        DigitPeaks(m,15) = DigitPeaks(m,14) + DigitPeaks(m+1,14) + DigitPeaks(m-1,14);
    end
end

```

```

elseif m == No_repart_subfrag
    DigitPeaks(m,15) = DigitPeaks(m,14) + DigitPeaks(m-1,14);
elseif m == 1
    DigitPeaks(m,15) = DigitPeaks(m,14) + DigitPeaks(m+1,14);
end
end
for m=1:No_repart_subfrag
    if m > 1 && m < No_repart_subfrag
        if DigitPeaks(m,15) > 4
            DigitPeaks(m,16) = 1;
            DigitPeaks(m+1, 16) = 1;
            DigitPeaks(m-1, 16) = 1;
        end
    elseif m == 1
        if DigitPeaks(m,15) > 4
            DigitPeaks(m,16) = 1;
            DigitPeaks(m+1,16) = 1;
        end
    elseif m == No_repart_subfrag
        if DigitPeaks(m,15) > 4
            DigitPeaks(m,16) = 1;
        end
    end
end
DigitPeaksFlag = DigitPeaks(:,16);
SubFragment_statistics_aux_backup = zeros(No_repart_subfrag, 15);
for m=1:No_repart_subfrag
    if DigitPeaksFlag(m,1) > 0
        SubFragment_statistics_aux_backup(m,:) = SubFragment_statistics_aux(m,:);
    end
end
SubFragment_statistics_aux_backup(~any(SubFragment_statistics_aux_backup,2), : ) = [];
SubFragment_statistics_aux = SubFragment_statistics_aux_backup;

%For plausible wall rearing, some fingertips shall demonstrate slow
%variations as it rest on wall, while others shall demonstrate mobility
No_repart_subfrag = size(SubFragment_statistics_aux,1);
for k=1:No_repart_subfrag
    count = 0;
    for j=1:No_of_labeled_points-1
        if SubFragment_statistics_aux(k,2+No_of_labeled_points+j) <= Variation_TH
            count = count + 1;
        end
    end
    SubFragment_statistics_aux(k,15) = count;
end

%Eliminate SubFragments of false positives for the following conditions
%1) Number of slow moving fingertips between 2 to 4
%2) Eliminate slow moving and fast moving palm
% Slow moving implies entire paw is resting
% Fast moving does not make posture sense

```



```

%3) Each fingertip cannot distant from each other more than 100 pixels
SubFragment_statistics_aux_tmp = SubFragment_statistics_aux;
for k=1:No_repart_subfrag
    clear_subfrag = 0;
    if (SubFragment_statistics_aux(k,15) < 2 || SubFragment_statistics_aux(k,15) > 4)
        %For wall rearing, the five fingertips has either 2,3, or 4
        %slow moving
        clear_subfrag = 1;
    end

    if int16(SubFragment_statistics_aux(k,15)) == 4
        %if four fingertips are slow moving, inspect whether the palm is
        %slow moving or ultra fast moving
        if SubFragment_statistics_aux(k,14) <= Variation_TH
            clear_subfrag = 1;
        elseif SubFragment_statistics_aux(k,14) >= Variation_High_TH
            clear_subfrag = 1;
        end
    end

    %Compute maxima and minima of slow moving fingertips
    slow_radix_matrix = zeros(int16(SubFragment_statistics_aux(k,15)),1);
    m=1;
    for j=1:(No_of_labeled_points-1)
        if SubFragment_statistics_aux(k,2+No_of_labeled_points + j) <= Variation_TH
            slow_radix_matrix(m) = SubFragment_statistics_aux(k,2+j);
            m=m+1;
        end
    end
    if max(slow_radix_matrix) - min(slow_radix_matrix) >= Max_radix_differentials
        clear_subfrag = 1;
    end

    %Identify the three slowest moving fingertips and remove those that lie
    %within Inner_Radius
    Variations = zeros(5,1);
    for j=9:13
        Variations(j-8,1) = SubFragment_statistics_aux(k,j);
    end
    Low_var = min(Variations,3);
    for j=3:7
        for m = 1:3
            if SubFragment_statistics_aux(k,j+No_of_labeled_points) == Low_var(m)
                if SubFragment_statistics_aux(k,j) < Inner_radius_outlier_removal_factor*Inner_Diameter ✓
/2
                    clear_subfrag = 1;
                end
            end
        end
    end
end

if clear_subfrag == 1

```



```

        for j=1:(3+2*No_of_labeled_points)
            SubFragment_statistics_aux_tmp(k,j) = 0;
        end
    end
end
%Remove all zeros rows data( ~any(data,2), : ) = []; %rows
SubFragment_statistics_aux_tmp( ~any(SubFragment_statistics_aux_tmp ,2),:) = [];

%Reconstruct SubFragment report
n=1;
No_repart_subfrag = size(SubFragment_statistics_aux_tmp,1);
for m=1:No_of_fragments
    Start_fragment = int16(Area_statistics(m,1));
    End_fragment = int16(Area_statistics(m,2));

    Fragment_data_warehouse_tmp2(n,1) = Start_fragment;
    Fragment_data_warehouse_tmp2(n,2) = End_fragment;
    n=n+1;

    for k=1:No_repart_subfrag
        Start_SubFragment = SubFragment_statistics_aux_tmp(k,1);
        End_SubFragment = SubFragment_statistics_aux_tmp(k,2);

        if (Start_SubFragment >= Start_fragment && Start_SubFragment < End_fragment) && End_SubFragment
            <= (End_fragment + Frame_Rate)
                Fragment_data_warehouse_tmp2(n,1) = int16((Start_SubFragment + End_SubFragment)/2);
                Fragment_data_warehouse_tmp2(n,2) = Start_SubFragment;
                Fragment_data_warehouse_tmp2(n,3) = End_SubFragment;
                Fragment_data_warehouse_tmp2(n,4) = 1;
                n = n+1;
            end
        end
    end
end
%Remove all zeros rows data( ~any(data,2), : ) = []; %rows
Fragment_data_warehouse_tmp2( ~any(Fragment_data_warehouse_tmp2 ,2),:) = [];

%-----%
%For fast moving SubFragments, Eliminate SubFragments of false positives for the following conditions
%1) Number of slow moving fingertips less than 2
%2) Identify three fingertips of slowest variations
%3) All fingertips has to lie beyond Inner Diameter
SubFragment_statistics_aux_tmp1 = SubFragment_statistics_aux;
No_repart_subfrag = size(SubFragment_statistics_aux,1);
for k=1:No_repart_subfrag
    clear_subfrag = 0;
    if SubFragment_statistics_aux(k,15) > 1
        %Eliminate all SubFragments that has more than 1 stable fingertips
        clear_subfrag = 1;
    end

    if clear_subfrag == 1
        for j=1:(3+2*No_of_labeled_points)

```

```

        SubFragment_statistics_aux_tmp1(k,j) = 0;
    end
end
end
%Remove all zeros rows data( ~any(data,2), : ) = []; %rows
SubFragment_statistics_aux_tmp1( ~any(SubFragment_statistics_aux_tmp1 ,2), :) = [];

No_repart_subfrag = size(SubFragment_statistics_aux_tmp1,1);
SubFragment_statistics_aux_tmp5 = SubFragment_statistics_aux_tmp1;
for k=1:No_repart_subfrag
    clear_subfrag = 0;

    %Remove SubFragments where more than 3 digits radii distance lie
    %within Inner_Radius

    No_of_inner_digits = 0;
    for j=3:8
        if SubFragment_statistics_aux_tmp5(k,j) < Inner_radius_outlier_removal_factor*Inner_Diameter /2
            No_of_inner_digits = No_of_inner_digits + 1;
            SubFragment_statistics_aux_flag(k,1) = No_of_inner_digits;
        end
    end
    if No_of_inner_digits > 3
        clear_subfrag = 1;
    end

    %Compute maxima and minima of slow moving fingertips
    slow_radii_matrix = zeros(int16(SubFragment_statistics_aux(k,15)),1);
    m=1;
    for j=1:(No_of_labeled_points-1)
        if SubFragment_statistics_aux(k,2+No_of_labeled_points + j) <= Variation_TH
            slow_radii_matrix(m) = SubFragment_statistics_aux(k,2+j);
            m=m+1;
        end
    end
    if max(slow_radii_matrix) - min(slow_radii_matrix) >= Max_radii_differentials
        clear_subfrag = 1;
    end

    if clear_subfrag == 1
        for j=1:(3+2*No_of_labeled_points)
            SubFragment_statistics_aux_tmp5(k,j) = 0;
        end
    end
end
end
%Remove all zeros rows data( ~any(data,2), : ) = []; %rows
SubFragment_statistics_aux_tmp5( ~any(SubFragment_statistics_aux_tmp5 ,2), :) = [];
SubFragment_statistics_aux_tmp1 = SubFragment_statistics_aux_tmp5;

%Reconstruct SubFragment report
n=1;

```

```

No_repart_subfrag = size(SubFragment_statistics_aux_tmp1,1);
for m=1:No_of_fragments
    Start_fragment = int16(Area_statistics(m,1));
    End_fragment = int16(Area_statistics(m,2));

    Fragment_data_warehouse_tmp3(n,1) = Start_fragment;
    Fragment_data_warehouse_tmp3(n,2) = End_fragment;
    n=n+1;

    for k=1:No_repart_subfrag
        Start_SubFragment = SubFragment_statistics_aux_tmp1(k,1);
        End_SubFragment = SubFragment_statistics_aux_tmp1(k,2);

        if (Start_SubFragment >= Start_fragment && Start_SubFragment < End_fragment) && End_SubFragment
            <= (End_fragment + Frame_Rate)
                Fragment_data_warehouse_tmp3(n,1) = int16((Start_SubFragment + End_SubFragment)/2);
                Fragment_data_warehouse_tmp3(n,2) = Start_SubFragment;
                Fragment_data_warehouse_tmp3(n,3) = End_SubFragment;
                Fragment_data_warehouse_tmp3(n,4) = 1;
                n = n+1;
            end
        end
    end
end
%Remove all zeros rows data( ~any(data,2), : ) = []; %rows
Fragment_data_warehouse_tmp3( ~any(Fragment_data_warehouse_tmp3 ,2),:) = [];

%***** Consolidate subfragments into episodes *****
Fragment_data_warehouse_tmp4 = zeros(1000,4);

%Eliminate zero rows of SubFragment statistics aux, and concatenate the two
%SubFragment statistics into one array
SubFragment_statistics_aux_tmp2 = SubFragment_statistics_aux_tmp;
SubFragment_statistics_aux_tmp3 = SubFragment_statistics_aux_tmp1;

SubFragment_statistics_aux_tmp2( ~any(SubFragment_statistics_aux_tmp2 ,2),:) = [];
SubFragment_statistics_aux_tmp3( ~any(SubFragment_statistics_aux_tmp3 ,2),:) = [];
SubFragment_statistics_aux_tmp4 = cat(1,SubFragment_statistics_aux_tmp2,
SubFragment_statistics_aux_tmp3);
SubFragment_statistics_aux_tmp4 = sort(SubFragment_statistics_aux_tmp4);

n=1;
No_repart_subfrag = size(SubFragment_statistics_aux_tmp4,1);
for m=1:No_of_fragments
    Start_fragment = int16(Area_statistics(m,1));
    End_fragment = int16(Area_statistics(m,2));

    Fragment_data_warehouse_tmp4(n,1) = Start_fragment;
    Fragment_data_warehouse_tmp4(n,2) = End_fragment;
    n=n+1;

    for k=1:No_repart_subfrag

```

```

Start_SubFragment = SubFragment_statistics_aux_tmp4(k,1);
End_SubFragment = SubFragment_statistics_aux_tmp4(k,2);

if Start_SubFragment >= Start_fragment && End_SubFragment <= End_fragment
    Fragment_data_warehouse_tmp4(n,1) = int16((Start_SubFragment + End_SubFragment)/2);
    Fragment_data_warehouse_tmp4(n,2) = Start_SubFragment;
    Fragment_data_warehouse_tmp4(n,3) = End_SubFragment;
    Fragment_data_warehouse_tmp4(n,4) = 1;
    n = n+1;
end
end
end
Fragment_data_warehouse_tmp4( ~any(Fragment_data_warehouse_tmp4 ,2),:) = [];

%***** Wall Rearing SubFragment Consolidation *****
%Time difference within which subfragments consolidate
Consolidate_SubFrag_time = 3;
%Minimum Episode Time Parameter refers to the least possible episode
%considered for reporting
Min_Episode_Time = 0.25;

Fragment_data_warehouse_tmp4 = zeros(1000,4);

%Eliminate zero rows of SubFragment statistics aux, and concatenate the two
%SubFragment statistics into one array
SubFragment_statistics_aux_tmp2 = SubFragment_statistics_aux_tmp;
SubFragment_statistics_aux_tmp3 = SubFragment_statistics_aux_tmp1;

SubFragment_statistics_aux_tmp2( ~any(SubFragment_statistics_aux_tmp2 ,2),:) = [];
SubFragment_statistics_aux_tmp3( ~any(SubFragment_statistics_aux_tmp3 ,2),:) = [];
SubFragment_statistics_aux_tmp4 = cat(1,SubFragment_statistics_aux_tmp2,
SubFragment_statistics_aux_tmp3);
SubFragment_statistics_aux_tmp4 = sort(SubFragment_statistics_aux_tmp4);

n=1;
No_repart_subfrag = size(SubFragment_statistics_aux_tmp4,1);
for m=1:No_of_fragments
    Start_fragment = int16(Area_statistics(m,1));
    End_fragment = int16(Area_statistics(m,2));

    Fragment_data_warehouse_tmp4(n,1) = Start_fragment;
    Fragment_data_warehouse_tmp4(n,2) = End_fragment;
    n=n+1;

    for k=1:No_repart_subfrag
        Start_SubFragment = SubFragment_statistics_aux_tmp4(k,1);
        End_SubFragment = SubFragment_statistics_aux_tmp4(k,2);

        if Start_SubFragment >= Start_fragment && End_SubFragment <= End_fragment
            Fragment_data_warehouse_tmp4(n,1) = int16((Start_SubFragment + End_SubFragment)/2);
            Fragment_data_warehouse_tmp4(n,2) = Start_SubFragment;
            Fragment_data_warehouse_tmp4(n,3) = End_SubFragment;

```



```

        Fragment_data_warehouse_tmp4(n,4) = 1;
        n = n+1;
    end
end
end
Fragment_data_warehouse_tmp4( ~any(Fragment_data_warehouse_tmp4 ,2), :) = [];

%***** Wall Rearing SubFragment Consolidation *****
%Consolidate SubFragments to Episodes, where successive subfragment are
%less than 2 seconds from each other
No_coherent_SubFragments = size(Fragment_data_warehouse_tmp4,1);
Fragment_data_warehouse_final = zeros(1,4);
SubFragment_data_warehouse = zeros(1,4);
SubFragment_data_warehouse_tmp = zeros(1,4);
Consol_SubFragments = zeros(No_coherent_SubFragments, 1);
Start_consolidation = 0;
for i=1:No_coherent_SubFragments
    if Fragment_data_warehouse_tmp4(i,3) > 0
        Start_consolidation = 1;
        Fragment_data_line(1,:) = Fragment_data_warehouse_tmp4(i,:);
        Fragment_data_line(1,4) = 1;
        SubFragment_data_warehouse = cat(1, SubFragment_data_warehouse, Fragment_data_line);
    else
        if Start_consolidation > 0
            SubFragment_data_warehouse( ~any(SubFragment_data_warehouse,2), : ) = [];
            Size_SubFragment_data_warehouse = size(SubFragment_data_warehouse, 1);
            for m=1:Size_SubFragment_data_warehouse
                if m == 1
                    Start_Subfragment = SubFragment_data_warehouse(m,2);
                end
                if m < Size_SubFragment_data_warehouse
                    if SubFragment_data_warehouse(m+1,2) - SubFragment_data_warehouse(m,3) <= 2 * Frame_Rate
                        End_Subfragment = SubFragment_data_warehouse(m+1,3);
                    else
                        End_Subfragment = SubFragment_data_warehouse(m,3);
                        Fragment_data_line(1,1) = int16((Start_Subfragment + End_Subfragment)/2);
                        Fragment_data_line(1,2) = Start_Subfragment;
                        Fragment_data_line(1,3) = End_Subfragment;
                        Fragment_data_line(1,4) = 1;

                        %Save if Subfragment duration > Minimum Episode
                        %duration
                        if (End_Subfragment - Start_Subfragment) > Min_Episode_Time * Frame_Rate
                            SubFragment_data_warehouse_tmp = cat(1, SubFragment_data_warehouse_tmp,
                                Fragment_data_line);
                        end

                        Start_Subfragment = SubFragment_data_warehouse(m+1,2);
                        Start_consolidation = 0;
                    end
                elseif m == Size_SubFragment_data_warehouse

```

```

End_Subfragment = SubFragment_data_warehouse(m,3);
Fragment_data_line(1,1) = int16((Start_Subfragment + End_Subfragment)/2);
Fragment_data_line(1,2) = Start_Subfragment;
Fragment_data_line(1,3) = End_Subfragment;
Fragment_data_line(1,4) = 1;

%Save if Subfragment duration > Minimum Episode
%duration
if (End_Subfragment - Start_Subfragment) > Min_Episode_Time * Frame_Rate
    SubFragment_data_warehouse_tmp = cat(1, SubFragment_data_warehouse_tmp, ✓
Fragment_data_line);
end
SubFragment_data_warehouse_tmp( ~any(SubFragment_data_warehouse_tmp,2), : ) = [];

Fragment_data_warehouse_final = cat(1, Fragment_data_warehouse_final, ✓
SubFragment_data_warehouse_tmp);

SubFragment_data_warehouse_tmp = zeros(1,4);
SubFragment_data_warehouse = zeros(1,4);

Start_consolidation = 0;
end
end
else
    SubFragment_data_warehouse = zeros(1,4);
end
end

if i == No_coherent_SubFragments %End of Fragment_data_warehouse_tmp4
    if Start_consolidation > 0
        SubFragment_data_warehouse( ~any(SubFragment_data_warehouse,2), : ) = [];
        Size_SubFragment_data_warehouse = size(SubFragment_data_warehouse, 1);
        for m=1:Size_SubFragment_data_warehouse
            if m == 1
                Start_Subfragment = SubFragment_data_warehouse(m,2);
            end
            if m < Size_SubFragment_data_warehouse
                if SubFragment_data_warehouse(m+1,2) - SubFragment_data_warehouse(m,3) <= 2 * ✓
Frame_Rate
                    End_Subfragment = SubFragment_data_warehouse(m+1,3);
                else
                    End_Subfragment = SubFragment_data_warehouse(m,3);
                    Fragment_data_line(1,1) = int16((Start_Subfragment + End_Subfragment)/2);
                    Fragment_data_line(1,2) = Start_Subfragment;
                    Fragment_data_line(1,3) = End_Subfragment;
                    Fragment_data_line(1,4) = 1;

                    %Save if Subfragment duration > Minimum Episode
                    %duration
                    if (End_Subfragment - Start_Subfragment) > Min_Episode_Time * Frame_Rate
                        SubFragment_data_warehouse_tmp = cat(1, SubFragment_data_warehouse_tmp, ✓

```



```

Fragment_data_line);
    end

    Start_Subfragment = SubFragment_data_warehouse(m+1,2);
    Start_consolidation = 0;
end
elseif m == Size_SubFragment_data_warehouse
    End_Subfragment = SubFragment_data_warehouse(m,3);
    Fragment_data_line(1,1) = int16((Start_Subfragment + End_Subfragment)/2);
    Fragment_data_line(1,2) = Start_Subfragment;
    Fragment_data_line(1,3) = End_Subfragment;
    Fragment_data_line(1,4) = 1;

    %Save if Subfragment duration > Minimum Episode
    %duration
    if (End_Subfragment - Start_Subfragment) > Min_Episode_Time * Frame_Rate
        SubFragment_data_warehouse_tmp = cat(1, SubFragment_data_warehouse_tmp, ✓
Fragment_data_line);
    end
    SubFragment_data_warehouse_tmp( ~any(SubFragment_data_warehouse_tmp,2), : ) = [];

    Fragment_data_warehouse_final = cat(1, Fragment_data_warehouse_final, ✓
SubFragment_data_warehouse_tmp);

    SubFragment_data_warehouse_tmp = zeros(1,4);
    SubFragment_data_warehouse = zeros(1,4);

    Start_consolidation = 0;
end
end
end
end
end
Fragment_data_warehouse_final( ~any(Fragment_data_warehouse_final,2), : ) = [];

%***** Dump Raw Data of final detected fragment *****
No_repart_subfrag = size(Fragment_data_warehouse_final,1);
RawTable_final = zeros(no_of_rows, columns);
for m=1:No_repart_subfrag
    Start_Subfragment = int16(Fragment_data_warehouse_final(m,2)-Frame_Rate*Consolidate_SubFrag_time);
    End_Subfragment = int16(Fragment_data_warehouse_final(m,3)+Frame_Rate*Consolidate_SubFrag_time);
    if Start_Subfragment < 1
        Start_Subfragment = 1;
    end

    if End_Subfragment > no_of_rows
        End_Subfragment = no_of_rows;
    end

    for i=Start_Subfragment:End_Subfragment
        RawTable_final(i,:) = RawTable(i,:);
    end
end
end
```

```

    end
end
%*****END of Fragment Raw Data *****

%**** Prepare episode summary report *****
n=1;
Fragment_data_warehouse_final_tmp = zeros(1,4);
No_repart_subfrag = size(Fragment_data_warehouse_final,1);
for m=1:No_of_fragments
    Start_fragment = int16(Area_statistics(m,1));
    End_fragment = int16(Area_statistics(m,2));

    Fragment_data_warehouse_final_tmp(n,1) = Start_fragment;
    Fragment_data_warehouse_final_tmp(n,2) = End_fragment;
    n=n+1;

    for k=1:No_repart_subfrag
        Start_SubFragment = Fragment_data_warehouse_final(k,2);
        End_SubFragment = Fragment_data_warehouse_final(k,3);

        if Start_SubFragment >= Start_fragment && End_SubFragment <= End_fragment
            Fragment_data_warehouse_final_tmp(n,1) = int16((Start_SubFragment + End_SubFragment)/2);
            Fragment_data_warehouse_final_tmp(n,2) = Start_SubFragment;
            Fragment_data_warehouse_final_tmp(n,3) = End_SubFragment;
            Fragment_data_warehouse_final_tmp(n,4) = 1;
            n = n+1;
        end
    end
end

Fragment_data_warehouse_final_tmp( ~any(Fragment_data_warehouse_final_tmp ,2),:) = [];
Fragment_data_warehouse_final = Fragment_data_warehouse_final_tmp;

%***** Output Files Specifications *****
warning('off','MATLAB:xlswrite:AddSheet');

Fragment_data_warehouse_final_tmp = zeros(1000,5);
Fragment_data_warehouse_size = size(Fragment_data_warehouse_final,1);

Fragment_serial_number = 1;
for i=1:Fragment_data_warehouse_size
    if Fragment_data_warehouse_final(i,3) < 1
        Fragment_data_warehouse_final_tmp(i,1) = Fragment_serial_number;

        Fragment_data_warehouse_final_tmp(i,2) = Fragment_data_warehouse_final(i,1);
        Fragment_data_warehouse_final_tmp(i,3) = Fragment_data_warehouse_final(i,2);

        Fragment_serial_number = Fragment_serial_number + 1;
    else
        Fragment_data_warehouse_final_tmp(i,1) = 0;
    end
end

```

```

        Fragment_data_warehouse_final_tmp(i,2) = Fragment_data_warehouse_final(i,1);
        Fragment_data_warehouse_final_tmp(i,3) = Fragment_data_warehouse_final(i,2);
        Fragment_data_warehouse_final_tmp(i,4) = Fragment_data_warehouse_final(i,3);
        Fragment_data_warehouse_final_tmp(i,5) = Fragment_data_warehouse_final(i,4);
    end
end
Fragment_data_warehouse_final_tmp( ~any(Fragment_data_warehouse_final_tmp,2), : ) = [];

report_header_1 = ["Fragment Number", "Start Fragment", "End Fragment"];
report_header_2 = ["", "SubFragment Midpoint", "Start SubFragment", "End_SubFragment", "Decision"];

if Process_left_paw == 1
    xlswrite(outputfile,report_header_1,"Left Paw Summary", "B1");
    xlswrite(outputfile,report_header_2,"Left Paw Summary", "B2");
    xlswrite(outputfile,Fragment_data_warehouse_final_tmp,"Left Paw Summary", "B3");

    xlswrite(outputfile,RawTable_final,"Left Paw Details", "A1");
else
    xlswrite(outputfile,report_header_1,"Right Paw Summary", "B1");
    xlswrite(outputfile,report_header_2,"Right Paw Summary", "B2");
    xlswrite(outputfile,Fragment_data_warehouse_final_tmp,"Right Paw Summary", "B3");

    xlswrite(outputfile,RawTable_final,"Right Paw Details", "A1");
end

clear;

end

%***** Functions *****

function mvt_symbol = mvtSymbol(prior,latter)

    if prior > 1
        if latter > 1
            mvt_symbol = 15;
        elseif 0 < latter && latter <=1
            mvt_symbol = 14;
        elseif -1 <= latter && latter <=0
            mvt_symbol = 12;
        else
            mvt_symbol = 13;
        end
    elseif 0 < prior && prior <= 1
        if latter > 1
            mvt_symbol = 11;
        elseif 0 < latter && latter <=1
            mvt_symbol = 10;
        elseif -1 <= latter && latter <=0
            mvt_symbol = 8;
        else

```

```

        mvt_symbol = 9;
    end
elseif -1 <= prior && prior <=0
    if latter > 1
        mvt_symbol = 3;
    elseif 0 < latter && latter <=1
        mvt_symbol = 2;
    elseif -1 <= latter && latter <=0
        mvt_symbol = -1;
    else
        mvt_symbol = 1;
    end
else
    if latter > 1
        mvt_symbol = 7;
    elseif 0 < latter && latter <=1
        mvt_symbol = 6;
    elseif -1 <= latter && latter <=0
        mvt_symbol = 4;
    else
        mvt_symbol = 5;
    end
end
end

function decision = touchDetect(start, finish, first, second, third, palm )

    palm_forward = 0;
    palm_rearing = 0;
    palm_fast_slow_forward = 0;
    palm_slow_slow_forward = 0;
    palm_fast_slow_retrace = 0;
    palm_slow_slow_retrace = 0;
    palm_slow_fast_forward = 0;

    first_forward = 0;
    first_rearing = 0;
    first_fast_slow_forward = 0;
    first_slow_slow_forward = 0;
    first_fast_slow_retrace = 0;
    first_slow_slow_retrace = 0;
    first_slow_fast_forward = 0;

    second_forward = 0;
    second_rearing = 0;
    second_fast_slow_forward = 0;
    second_slow_slow_forward = 0;
    second_fast_slow_retrace = 0;
    second_slow_slow_retrace = 0;
    second_slow_fast_forward = 0;

    third_forward = 0;

```

```

third_rearing = 0;
third_fast_slow_forward = 0;
third_slow_slow_forward = 0;
third_fast_slow_retrace = 0;
third_slow_slow_retrace = 0;
third_slow_fast_forward = 0;

switch palm
    %Detect Palm rearing pattern before and after coherent subfragments
    case 15
        palm_forward = 1;
    case 14
        palm_fast_slow_forward = 1;
    case 13
        palm_rearing = 1;
    case 12
        palm_rearing = 1;
        palm_fast_slow_retrace = 1;
    case 9
        palm_rearing = 1;
    case 8
        palm_rearing = 1;
        palm_slow_slow_retrace = 1;
    otherwise
        palm_forward = 0;
        palm_rearing = 0;
        palm_fast_slow_forward = 0;
        palm_fast_slow_retrace = 0;
        palm_slow_slow_retrace = 0;
end

```

```

switch first
    case 15
        first_forward = 1;
    case 14
        first_fast_slow_forward = 1;
    case 13
        first_rearing = 1;
    case 12
        first_rearing = 1;
        first_fast_slow_retrace = 1;
    case 11
        first_slow_fast_forward = 1;
    case 10
        first_slow_slow_forward = 1;
    case 9
        first_rearing = 1;
        first_slow_fast_retrace = 1;
    case 8
        first_rearing = 1;
        first_slow_slow_retrace = 1;

```

```

otherwise
    first_forward = 0;
    first_rearing = 0;
    first_fast_slow_forward = 0;
    first_slow_slow_forward = 0;
    first_fast_slow_retrace = 0;
    first_slow_slow_retrace = 0;
    first_slow_fast_forward = 0;
end

switch second
case 15
    second_forward = 1;
case 14
    second_fast_slow_forward = 1;
case 13
    second_rearing = 1;
case 12
    second_rearing = 1;
    second_fast_slow_retrace = 1;
case 11
    second_slow_fast_forward = 1;
case 10
    second_slow_slow_forward = 1;
case 9
    second_rearing = 1;
    second_slow_fast_retrace = 1;
case 8
    second_rearing = 1;
    second_slow_slow_retrace = 1;
otherwise
    second_forward = 0;
    second_rearing = 0;
    second_fast_slow_forward = 0;
    second_slow_slow_forward = 0;
    second_fast_slow_retrace = 0;
    second_slow_slow_retrace = 0;
    second_slow_fast_forward = 0;
end

switch third
case 15
    third_forward = 1;
case 14
    third_fast_slow_forward = 1;
case 13
    third_rearing = 1;
case 12
    third_rearing = 1;
    third_fast_slow_retrace = 1;
case 11

```



```

        third_slow_fast_forward = 1;
    case 10
        third_slow_slow_forward = 1;
    case 9
        third_rearing = 1;
        third_slow_fast_retrace = 1;
    case 8
        third_rearing = 1;
        third_slow_slow_retrace = 1;
    otherwise
        third_forward = 0;
        third_rearing = 0;
        third_fast_slow_forward = 0;
        third_slow_slow_forward = 0;
        third_fast_slow_retrace = 0;
        third_slow_slow_retrace = 0;
        third_slow_fast_forward = 0;
    end

    if (palm_rearing + first_rearing + second_rearing + third_rearing) >= 2
        %2 or more finger / palm demonstrate rearing pattern
        decision = 1;
    elseif (palm_rearing + first_rearing + second_rearing + third_rearing) > 0 && (palm_rearing +
first_rearing + second_rearing + third_rearing) < 2
        decision = 0.5;
    else
        decision = 0;
    end
end
end

```

```

%Parameter Script
function FunctionPalm_RightRefine()
%***** Read PalmGrid Parameter Files *****%

inputfile = "G:\Tim's Lab\EMXMas\PalmGridParam.xlsx";

delimiter=' ';
PalmGridParam_struct =importdata(inputfile,delimiter,1);

%***** Step 1: Input Files Specifications *****
pc_cutoff = 0.1; % cutoff probability of predicted coordinates
Frame_Rate = 25; %Video recorded in 25 frames per second
Center_X = PalmGridParam_struct.data.PalmGridSheet(1,1); %Cartesian Coordinates of Center (X,Y) of
Cylinder
Center_Y = PalmGridParam_struct.data.PalmGridSheet(2,1);
Inner_Diameter = PalmGridParam_struct.data.PalmGridSheet(3,1); %Inner diameter of cylinder in pixels

filename = char(PalmGridParam_struct.textdata.PalmGridSheet(1,1));
video = char(PalmGridParam_struct.textdata.PalmGridSheet(2,1));

%***** Step 2: Smoothing Specifications *****
LabeledPointsPerPaw = 6; %Number of labeled points per paw
Smooth_window = 24; %Size of time window for initial signal smoothing

%***** Step 3: Left / Right Paw Polar Conversion *****
Peak_Threshold = 3; %All data within 2 pixels of local peaks are excluded

%***** Step 4: Extract Congruent Zones Specifications *****
TimeWindow = 6; %Specify time window of smoothing
Threshold = 5; %Specify threshold of screening within 5 pixels
No_of_labeled_points = 6; %Specify number of labeled points per paw
Min_Slow_Movement_Frames = 3; %Minimal Number of consecutive slow movement frames that paw shall be
held before considering a contact
Min_Flag_To_Stay = 0; %Minimal Number of stable points detected in any particular frame to be
considered in Congruence Zone
area_statistics_columns = 13;

%***** Step 5: Extract Coherent Subfragment Specifications *****
Max_Palm_Threshold = 8; %Palm that moves beyond 8 pixels per frame shall be considered fast movement
Min_Palm_Threshold = 5; %Palm that moves below 5 pixels per frame shall be considered slow movement
Stationary_Mvt_Threshold = 1; %Average Movement of less than 1 pixels per frame is considered
stationary movements
safety_margin = 0.85; %Safety margin of Inner Diameter, within which paw most likely not wall rearing

Min_Congruence = 2; %MC=2; Minimum number of congruence points required to count as congruent intervals
Max_SubFragment_Gap = 2; %Maximum number of frames that two subfragments can be separated in order to
be considered merging two subfragments as one
Congruence_window = 8; %CW = 6; Number of frames (hence time windows) within which minima of FingerTips
- Palm are coherently reached
Inspection_window = 12; %IW = 12; Time Window within which congruence shall be inspected
Columns_of_SubFragments = 11;

```

```

%***** Step 6: Extract Congruent Points Specifications *****

%***** Step 7: Decisions Evaluation Specifications *****
Fragment_ref = 0;
Crawling_low_threshold = 10;
Crawling_high_threshold = 100;
Minimum_touch_duration = Frame_Rate * 0.1; % At least 0.1 seconds of slow movements to classify as ↙
stable touch

Min_stat_computation = 3; %Minimum time frames below which repartitioned subfragment will not be ↙
computed of statistics

Variation_TH = 13;
Variation_High_TH = 100;
Max_radii_differentials = 120;
Inner_radius_outlier_removal_factor = 1.0;

Transition_TH = 100; %change in number of pixels between frame exceed 100 as transition threshold

%***** Step 8: Consolidate subfragments into episodes *****
%Time difference within which subfragments consolidate
Consolidate_SubFrag_time = 3;
%Minimum Episode Time Parameter refers to the least possible episode
%considered for reporting
Min_Episode_Time = 0.25;

%***** Step 9: Output Files Specifications *****
outputfile = char(PalmGridParam_struct.textdata.PalmGridSheet(3,1));

%***** Input Files Specifications *****
%Process Left Paw
Process_left_paw = 0;

RawTable=csvread(filename,3,1);
[no_of_rows, columns]=size(RawTable);

%Initialize Left Front Paw and Right Front Paw Arrays
Lfp = zeros(no_of_rows, 12);
Lfp_P = zeros(no_of_rows, 7);
Rfp = zeros(no_of_rows, 12);
Rfp_P = zeros(no_of_rows, 7);

for i=1:no_of_rows %Assign RawTable to Tables Head/Lfp/Rfp/Lbp/Rbp, and evaluate Likelihood
%Lfp1 = Left Front Paw's Thumb
%Lfp2 = Left Front Paw's Index finger
%Lfp3 = Left Front Paw's Long finger
%Lfp4 = Left Front Paw's Ring finger

```

```

%Lfp5 = Left Front Paw's Small finger
%Lfp6 = Left Front Paw's Radius

Lfp(i,1)=RawTable(i,1); %Lfp1_x
Lfp(i,2)=RawTable(i,2); %Lfp1_y
Lfp(i,3)=RawTable(i,4); %Lfp2_x
Lfp(i,4)=RawTable(i,5); %Lfp2_y
Lfp(i,5)=RawTable(i,7); %Lfp3_x
Lfp(i,6)=RawTable(i,8); %Lfp3_y
Lfp(i,7)=RawTable(i,10); %Lfp4_x
Lfp(i,8)=RawTable(i,11); %Lfp4_y
Lfp(i,9)=RawTable(i,13); %Lfp5_x
Lfp(i,10)=RawTable(i,14); %Lfp5_y
Lfp(i,11)=RawTable(i,16); %Lfp6_x
Lfp(i,12)=RawTable(i,17); %Lfp6_y

Lfp_P(i,1)=RawTable(i,3); %Likelihood Lfp1
Lfp_P(i,2)=RawTable(i,6); %Likelihood Lfp2
Lfp_P(i,3)=RawTable(i,9); %Likelihood Lfp3
Lfp_P(i,4)=RawTable(i,12); %Likelihood Lfp4
Lfp_P(i,5)=RawTable(i,15); %Likelihood Lfp5
Lfp_P(i,6)=RawTable(i,18); %Likelihood Lfp6

Reliability = 6; %Compute Reliability of Coordinates in Head
for j=1:6
    if Lfp_P(i,j) < pc_cutoff
        Reliability=Reliability-1;
    end
end
Lfp_P(i,7)=Reliability;

%Rfp1 = Right Front Paw's Thumb
%Rfp2 = Right Front Paw's Index finger
%Rfp3 = Right Front Paw's Long finger
%Rfp4 = Right Front Paw's Ring finger
%Rfp5 = Right Front Paw's Small finger
%Rfp6 = Right Front Paw's Radius

Rfp(i,1)=RawTable(i,19); %Rfp1_x
Rfp(i,2)=RawTable(i,20); %Rfp1_y
Rfp(i,3)=RawTable(i,22); %Rfp2_x
Rfp(i,4)=RawTable(i,23); %Rfp2_y
Rfp(i,5)=RawTable(i,25); %Rfp3_x
Rfp(i,6)=RawTable(i,26); %Rfp3_y
Rfp(i,7)=RawTable(i,28); %Rfp4_x
Rfp(i,8)=RawTable(i,29); %Rfp4_y
Rfp(i,9)=RawTable(i,31); %Rfp3_x
Rfp(i,10)=RawTable(i,32); %Rfp3_y
Rfp(i,11)=RawTable(i,34); %Rfp4_x
Rfp(i,12)=RawTable(i,35); %Rfp4_y

Rfp_P(i,1)=RawTable(i,21); %Likelihood Rfp_1

```



```

Rfp_P(i,2)=RawTable(i,24); %Likelihood Rfp_2
Rfp_P(i,3)=RawTable(i,27); %Likelihood Rfp_3
Rfp_P(i,4)=RawTable(i,30); %Likelihood Rfp_4
Rfp_P(i,5)=RawTable(i,33); %Likelihood Rfp_3
Rfp_P(i,6)=RawTable(i,36); %Likelihood Rfp_4

Reliability = 6; %Compute Reliability of Coordinates in Left Front Paw
for j=1:6
    if Rfp_P(i,j) < pc_cutoff
        Reliability=Reliability-1;
    end
end
Rfp_P(i,7)=Reliability;

end

%***** Smoothing Specifications *****
%Smooth the predicted coordinates by averaging +/- 12 extracted coordinates
half_Smooth_window = Smooth_window / 2;

buffer = zeros(no_of_rows, 2*LabeledPointsPerPaw);
for i = 1:half_Smooth_window %Smooth Left Front Paw for first few no_of_rows
    for j= 1:2*LabeledPointsPerPaw
        buffer (i,j) = Lfp (i,j);
    end
end
for i = half_Smooth_window:(no_of_rows-half_Smooth_window) %Beyond the first few no_of_rows, Lfp
coordinates are smoothed based on past Smoothed_window records
    for j= 1:2*LabeledPointsPerPaw
        Avg_coor = 0.00;
        for k = 0:half_Smooth_window
            Avg_coor = Avg_coor + (Lfp (i-k+1,j) + Lfp(i+k,j))/((k+1)/2);
        end
        buffer (i,j) = Avg_coor / Smooth_window;
    end
end
for i = (no_of_rows-half_Smooth_window):no_of_rows %Smooth Left Front Paw for last few no_of_rows
    for j= 1:2*LabeledPointsPerPaw
        buffer (i,j) = Lfp (i,j);
    end
end
filteredLfp = buffer;

for i = 1:half_Smooth_window %Smooth Right Front Paw for first few no_of_rows
    for j= 1:2*LabeledPointsPerPaw
        buffer (i,j) = Rfp (i,j);
    end
end
for i = half_Smooth_window:(no_of_rows-half_Smooth_window) %Beyond the first few no_of_rows, Rfp
coordinates are smoothed based on past Smoothed_window records
    for j= 1:2*LabeledPointsPerPaw

```

```

    Avg_coor = 0.00;
    for k = 0:half_Smooth_window
        Avg_coor = Avg_coor + (Rfp (i-k+1,j) + Rfp(i+k,j))/((k+1)/2);
    end
    buffer (i,j) = Avg_coor / Smooth_window;
end
end
for i = (no_of_rows-half_Smooth_window):no_of_rows %Smooth Right Front Paw for last few no_of_rows
    for j= 1:2*LabeledPointsPerPaw
        buffer (i,j) = Rfp (i,j);
    end
end
filteredRfp = buffer;

%***** Left / Right Paw Polar Conversion *****

Rfp_Polar_Radius = zeros(no_of_rows,6);
Rfp_Polar_Angles = zeros(no_of_rows,6);
Inspect_Range = floor(Frame_Rate/4);
Minimum_Inspection_Window = 6;

for i=1:no_of_rows

    %Translate five Right Front Paw fingertips' cartesian to polar coordinates
    [Rfp_Polar_Angles(i,1),Rfp_Polar_Radius(i,1)]= cart2pol(filteredRfp(i,1)-Center_X,filteredRfp(i,2)-
Center_Y);
    [Rfp_Polar_Angles(i,2),Rfp_Polar_Radius(i,2)]= cart2pol(filteredRfp(i,3)-Center_X,filteredRfp(i,4)-
Center_Y);
    [Rfp_Polar_Angles(i,3),Rfp_Polar_Radius(i,3)]= cart2pol(filteredRfp(i,5)-Center_X,filteredRfp(i,6)-
Center_Y);
    [Rfp_Polar_Angles(i,4),Rfp_Polar_Radius(i,4)]= cart2pol(filteredRfp(i,7)-Center_X,filteredRfp(i,8)-
Center_Y);
    [Rfp_Polar_Angles(i,5),Rfp_Polar_Radius(i,5)]= cart2pol(filteredRfp(i,9)-Center_X,filteredRfp(i,10)-
Center_Y);
    [Rfp_Polar_Angles(i,6),Rfp_Polar_Radius(i,6)]= cart2pol(filteredRfp(i,11)-Center_X,filteredRfp(i,12)-
-Center_Y);

end

Polar_Radius = Rfp_Polar_Radius;

Rfp_Polar_Radius = zeros(no_of_rows,6);
Rfp_Polar_Angles = zeros(no_of_rows,6);

for i=1:no_of_rows

    %Translate five Left Front Paw fingertips' cartesian to polar coordinates
    [Rfp_Polar_Angles(i,1),Rfp_Polar_Radius(i,1)]= cart2pol(Rfp(i,1)-Center_X,Rfp(i,2)-Center_Y);
    [Rfp_Polar_Angles(i,2),Rfp_Polar_Radius(i,2)]= cart2pol(Rfp(i,3)-Center_X,Rfp(i,4)-Center_Y);
    [Rfp_Polar_Angles(i,3),Rfp_Polar_Radius(i,3)]= cart2pol(Rfp(i,5)-Center_X,Rfp(i,6)-Center_Y);
    [Rfp_Polar_Angles(i,4),Rfp_Polar_Radius(i,4)]= cart2pol(Rfp(i,7)-Center_X,Rfp(i,8)-Center_Y);

```



```

[Rfp_Polar_Angles(i,5),Rfp_Polar_Radius(i,5)]= cart2pol(Rfp(i,9)-Center_X,Rfp(i,10)-Center_Y);
[Rfp_Polar_Angles(i,6),Rfp_Polar_Radius(i,6)]= cart2pol(Rfp(i,11)-Center_X,Rfp(i,12)-Center_Y);

end

Raw_Polar_Radius = Rfp_Polar_Radius;

%***** Extract Congruent Zones Specifications *****
%Initialize working arrays
Average_Polar_Radius = zeros(no_of_rows, No_of_labeled_points*2);
Congruence_row = zeros(no_of_rows,1);

%Average Radial Displacements of each labeled points to extract principal
%axes
Average_Factor = 2 * TimeWindow + 1;
for i= (TimeWindow + 1): (no_of_rows - TimeWindow)
    for j=1:6
        Average_Pr = 0;
        for k = (i-TimeWindow):(i+TimeWindow)
            Average_Pr = Average_Pr + Polar_Radius(k,j);
        end
        Average_Pr = Average_Pr / Average_Factor;
        Average_Polar_Radius(i,j) = Average_Pr;
        Average_Polar_Radius(i,j+No_of_labeled_points) = Average_Polar_Radius(i,j) - ✓
    end
end
Average_Polar_Radius(i-1,j);

end

%Filter no_of_rows whose average radial displacement of either labeled points are
%beyond Threshold
Average_Polar_Radius_templ = Average_Polar_Radius;
for i= (TimeWindow + 1): (no_of_rows - TimeWindow)
    Flag_to_stay = 0; % Inspect all ?radius within threshold
    for j=No_of_labeled_points+1:No_of_labeled_points*2
        if (abs(Average_Polar_Radius_templ(i,j)) < Threshold && abs(Average_Polar_Radius_templ(i,j)) > ✓
0)
            %If any labeled points difference from last labeled points <
            %Threshold, flag the row to stay
            Flag_to_stay = 1;
        end
    end
    %If all labeled points radial displacement > Threshold, paw is likely
    %moving => unlikely to touch wall / very slow locomotion / resting on
    %floor
    if Flag_to_stay == 0
        for j=1:No_of_labeled_points * 2
            Average_Polar_Radius_templ(i,j)=0;
        end
    else
        Congruence_row (i) = 1;
    end
end
end

```

```

%Filter Congruence no_of_rows for at least 2 out of 5 finger tips demonstrate
%slow movements
Average_Polar_Radius_temp2=Average_Polar_Radius_temp1;
for i=1:no_of_rows
    if Congruence_row(i) > 0
        Flag_to_stay = 0;
        for j= (No_of_labeled_points + 1):(No_of_labeled_points*2 - 1)
            if abs(Average_Polar_Radius_temp2(i,j)) < Threshold && abs(Average_Polar_Radius_temp2(i,j)) <
                > 0
                    %Count number of finger tips that demonstrate slow movements
                    %between frames
                    Flag_to_stay = Flag_to_stay + 1;
                end
            end
        if Flag_to_stay < Min_Flag_To_Stay
            %if 1 out of 5 fingertips demonstrate movement < (Threshold; currently = 5)
            %pixels between successive frames; the frame can stay
            for j=1:No_of_labeled_points * 2
                Average_Polar_Radius_temp2(i,j) = 0;
                Congruence_row(i) = 0;
            end
        end
    end
end

%Eliminate discrete no_of_rows where consecutive congruence frames <
%(Min_Slow_Movement_Frames = 4)
Average_Polar_Radius_temp3 = zeros(no_of_rows, No_of_labeled_points*2);
for i=1:no_of_rows-1
    if(Congruence_row(i,1) < 1 && Congruence_row(i+1,1) >= 1)
        %transition to start of inspection fragment
        Start_fragment = i+1;
    elseif (Congruence_row(i,1) >= 1 && Congruence_row(i+1,1) < 1)
        %transition to end of inspection fragment
        End_fragment = i;
        if (End_fragment - Start_fragment + 1) >= Min_Slow_Movement_Frames
            for k=Start_fragment:End_fragment
                for j=1:No_of_labeled_points*2
                    Average_Polar_Radius_temp3(k,j) = Average_Polar_Radius_temp2(k,j);
                end
            end
        else
            for k=Start_fragment:End_fragment
                Congruence_row(k,1)=0;
            end
        end
    end
end
Average_Polar_Radius_temp2 = Average_Polar_Radius_temp3;

%Eliminate fragments where all Palms are moving in one direction

```

```

k=1;
palm_positive = 0;
palm_negative = 0;
Area_statistics = zeros(100,area_statistics_columns);
for i=1:no_of_rows-1
    if(Congruence_row(i,1) < 1 && Congruence_row(i+1,1) >= 1)
        %transition to start of inspection fragment
        Start_fragment = i+1;
    elseif (Congruence_row(i,1) >= 1 && Congruence_row(i+1,1) < 1)
        %transition to end of inspection fragment
        End_fragment = i;

        Area_statistics(k,1) = Start_fragment;
        Area_statistics(k,2) = End_fragment;

        if Average_Polar_Radius_temp2(i, No_of_labeled_points * 2) >= 0
            palm_positive = palm_positive + 1;
        else
            palm_negative = palm_negative + 1;
        end

        Area_statistics(k,3) = palm_positive;
        Area_statistics(k,4) = palm_negative;
        k = k+1;
    else
        if Average_Polar_Radius_temp2(i, No_of_labeled_points * 2) >= 0
            palm_positive = palm_positive + 1;
        else
            palm_negative = palm_negative + 1;
        end
    end
end
Area_statistics(~any(Area_statistics,2), : ) = [];

No_of_fragments = size(Area_statistics,1);
Average_Polar_Radius_temp3 = Average_Polar_Radius_temp2;
m = 1;
for k=1:No_of_fragments
    %if fragment either all moving forward or moving backward
    %eliminate fragment from wall rearing evaluation
    if(Area_statistics(k,3) == 0 || Area_statistics(k,4) == 0)
        Start_fragment = Area_statistics(k,1);
        End_fragment = Area_statistics(k,2);
        for i=Start_fragment:End_fragment
            for j=1:No_of_labeled_points * 2
                Average_Polar_Radius_temp3(i,j) = 0;
            end
            Congruence_row(i,1)=0;
        end
    else
        Area_statistics_temp(m,:) = Area_statistics(k,:);
        m = m + 1;
    end
end

```

```

    end
end
Area_statistics = Area_statistics_temp;
Average_Polar_Radius_temp2 = Average_Polar_Radius_temp3;
No_of_fragments = size(Area_statistics,1);

Average_Polar_Radius_temp3 = Average_Polar_Radius_temp2;
Fragment_Polar_Radius = zeros(no_of_rows,No_of_labeled_points * 3);
for k=1:No_of_fragments
    %Acquire start and end of fragment, and compute its duration
    Start_fragment = Area_statistics(k,1);
    End_fragment = Area_statistics(k,2);
    Duration_of_fragment = End_fragment - Start_fragment +1;

    for i=Start_fragment:End_fragment
        for j=1:No_of_labeled_points
            %Compute differential radial distance between (FingerTips to
            %Palm) to Fragment Array
            Fragment_Polar_Radius(i,j+No_of_labeled_points) = Average_Polar_Radius_temp3(i,
j+No_of_labeled_points);
            Fragment_Polar_Radius(i,j) = Average_Polar_Radius_temp3(i,j)-Average_Polar_Radius_temp3(i,
No_of_labeled_points);
        end
    end
    for m=Start_fragment:End_fragment
        %Flag all fingertips where differential radial distance are less
        %than threshold
        Fragment_Polar_Radius(m,3*No_of_labeled_points) = 0;
        for j = 1:No_of_labeled_points-1
            if abs(Fragment_Polar_Radius(m, j+No_of_labeled_points))< Threshold && abs
(Fragment_Polar_Radius(m, j+No_of_labeled_points)) > 0
                Fragment_Polar_Radius(m,j+2*No_of_labeled_points) = 1;
                Fragment_Polar_Radius(m,3*No_of_labeled_points) = Fragment_Polar_Radius(m,
3*No_of_labeled_points) + 1;
            else
                Fragment_Polar_Radius(m,j+2*No_of_labeled_points) = 0;
            end
        end
    end
end
end

%***** Extract Coherent Subfragment Specifications *****

%Initialize wall rearing fragment table
Fragment_data_warehouse = zeros(1,Columns_of_SubFragments);
Fragment_data_aux = zeros(1,17);

for k=1:No_of_fragments

```

```

%1) For each fragment, further decompose into SubFragments based on three
%most stable fingertips moving less than Min_palm_threshold;
%
%2) Followed by consolidation of SubFragments where consecutive
%subfragments are only separated by 1 frame, to avoid unnecessary
%subfragmentations due to minor obstructions;
%
%3) Compute statistics +/- 0.5 seconds around the consolidated
%subfragments, to determine what the particular paw did prior to its
%slow movements. Hence discern of wall rearing / floor rearing
%activities

Start_fragment = Area_statistics(k,1);
End_fragment = Area_statistics(k,2);
Duration_of_fragment = End_fragment - Start_fragment +1;

%***** Step 1 *****
%1) Identify the top three fingertips that demonstrate stability, and
%   record its start and end sub_fragment timings;

%2) Identify the stable Palm position within the sub_fragment timing
%   where consecutive radial distance are less than Max_Palm_Threshold;
%
%3) Compute the average radial distance of the finger tips and the palm;
%

%Step 1a): Identify Stable Fingertips
paw = zeros(No_of_labeled_points-1,2);
paw(1,2) = 1; % 1 = Thumb finger
paw(2,2) = 2; % 2 = index finger
paw(3,2) = 3; % 3 = Middle finger
paw(4,2) = 4; % 4 = Ring finger
paw(5,2) = 5; % 5 = small finger

Sum_of_minFingerPalm = 0;
Sum_of_maxFingerPalm = 0;
Average_stable_1st = 0;
Average_stable_2nd = 0;
Average_stable_3rd = 0;
Average_palm = 0;

for i=Start_fragment:End_fragment
    %Compute for individual finger how many frames demonstrate slow
    %movements ie ?(radial distance) < Max_Palm_Threshold
    for j = 2*No_of_labeled_points+1:2*No_of_labeled_points+5
        if Fragment_Polar_Radius(i,j) > 0
            switch j
                case 2*No_of_labeled_points+1
                    paw(1,1) = paw(1,1) + 1;
                case 2*No_of_labeled_points+2
                    paw(2,1) = paw(2,1) + 1;
                case 2*No_of_labeled_points+3

```



```

        paw(3,1) = paw(3,1) + 1;
    case 2*No_of_labeled_points+4
        paw(4,1) = paw(4,1) + 1;
    otherwise
        paw(5,1) = paw(5,1) + 1;
    end
end
end
%Identify the top 3 fingertips that are most stable within the
%fragment
sort_paw = sortrows(paw, 'descend');
stable_1st = int8(sort_paw(1,2));
stable_2nd = int8(sort_paw(2,2));
stable_3rd = int8(sort_paw(3,2));

Average_stable_1st = Average_stable_1st + Average_Polar_Radius_temp3(i,stable_1st);
Average_stable_2nd = Average_stable_2nd + Average_Polar_Radius_temp3(i,stable_2nd);
Average_stable_3rd = Average_stable_3rd + Average_Polar_Radius_temp3(i,stable_3rd);
Palm_rad(ii-Start_fragment+1)=Average_Polar_Radius_temp3(i,6);
Average_palm = Average_palm + Average_Polar_Radius_temp3(i,6);
end

Area_statistics(k,4+stable_1st) = Average_stable_1st / Duration_of_fragment;
Area_statistics(k,4+stable_2nd) = Average_stable_2nd / Duration_of_fragment;
Area_statistics(k,4+stable_3rd) = Average_stable_3rd / Duration_of_fragment;
Area_statistics(k,10) = Average_palm / Duration_of_fragment;

%Step 1b) Parse Stable FingerTips information to working array
Fragment_Polar_Radius_tmp = zeros(Duration_of_fragment, 6*No_of_labeled_points);
for i=Start_fragment:End_fragment
    for j = 1:No_of_labeled_points
        switch j
            case stable_1st
                Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
            case stable_2nd
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j+1) = ✓
            case stable_3rd
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j+1) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j+2) = ✓
            case stable_4th
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j+1) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j+2) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j+3) = ✓
            case stable_5th
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j+1) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j+2) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j+3) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j+4) = ✓
            case stable_6th
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,6*No_of_labeled_points+j) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,6*No_of_labeled_points+j+1) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,6*No_of_labeled_points+j+2) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,6*No_of_labeled_points+j+3) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,6*No_of_labeled_points+j+4) = ✓
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,6*No_of_labeled_points+j+5) = ✓
        end
    end
end
%Threshold
% (FingerTip - Palm distance moves less than 5
% pixels per frame
Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 1;
else
    Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 0;
end
case stable_2nd

```



```

        Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,No_of_labeled_points+j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,2*No_of_labeled_points+j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j)-Fragment_Polar_Radius(i-1,j);
        if abs(Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j)) < ✓
Threshold
            %(FingerTip - Palm distance moves less than 5
            %pixels per frame
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 1;
        else
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 0;
        end
    case stable_3rd
        Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,No_of_labeled_points+j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,2*No_of_labeled_points+j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j)-Fragment_Polar_Radius(i-1,j);
        if abs(Fragment_Polar_Radius_tmp(i-Start_fragment+1,4*No_of_labeled_points+j)) < ✓
Threshold
            %(FingerTip - Palm distance moves less than 5
            %pixels per frame
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 1;
        else
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points+j) = 0;
        end
    case No_of_labeled_points
        Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,No_of_labeled_points+j);
    end
end
    %Tag frames that exhibit coherent slow movement in FingerTips -
    %Palm distances
    Total_slow_mvt_pt = 0;
    for j=1:5
        Total_slow_mvt_pt = Total_slow_mvt_pt + Fragment_Polar_Radius_tmp(i-Start_fragment+1, ✓
5*No_of_labeled_points+j);
    end
    if Total_slow_mvt_pt >= Min_Congruence
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points) = 1;

```

```

else
    Fragment_Polar_Radius_tmp(i-Start_fragment+1,5*No_of_labeled_points) = 0;
end
end

%
%During wall-rearing, fingertips will demonstrate temporal stability,
%that FingerTips - Palm shall exhibit short period of small changes
%between frames < Minimum Movement Threshold
%
%2a)Gauge sub-fragments within the fragment where palm exhibit
% small changes in (FingerTips - Palm distance);
%
%2b)Consolidate the two sub-fragments into one sub-fragment array before
% proceeding to Step 3)

%----- Step 2a -----%
%Step 2a) Gauge subfragments that show coherent minimal changes in
% (FingerTips - Palm) distance

No_of_SubFragments = 100;
SubFrag_counter = 1;
SubFragment_stat_prior = zeros(No_of_SubFragments, 3*No_of_labeled_points+4);

Start_SubFragment = 1;
End_SubFragment = 1;
SubFragment_stat = zeros(No_of_SubFragments,Columns_of_SubFragments);

for m=1:Duration_of_fragment
    if m <= Duration_of_fragment-1
        %Within fragment treatment
        if Fragment_Polar_Radius_tmp(m,5*No_of_labeled_points) < 1 && Fragment_Polar_Radius_tmp
(m+1,5*No_of_labeled_points) == 1
            Start_SubFragment = m+1;
        elseif Fragment_Polar_Radius_tmp(m,5*No_of_labeled_points) == 1 &&
Fragment_Polar_Radius_tmp(m+1,5*No_of_labeled_points) < 1
            End_SubFragment = m;
            SubFragment_stat_prior(SubFrag_counter,1) = round((Start_SubFragment + End_SubFragment)
/2);

            SubFragment_stat_prior(SubFrag_counter,2) = Start_SubFragment;
            SubFragment_stat_prior(SubFrag_counter,3) = End_SubFragment;

            SubFrag_counter = SubFrag_counter + 1;
            Start_SubFragment = 1;
            End_SubFragment = 1;
        end
    else
        %End of fragment treatment where the last frame has coherence
        if Fragment_Polar_Radius_tmp(m,5*No_of_labeled_points) == 1
            End_SubFragment = Duration_of_fragment;
            SubFragment_stat_prior(SubFrag_counter,1) = round((Start_SubFragment + End_SubFragment)
/2);

```

```

        SubFragment_stat_prior(SubFrag_counter,2) = Start_SubFragment;
        SubFragment_stat_prior(SubFrag_counter,3) = End_SubFragment;
    end

    SubFrag_counter = 1;
    Start_SubFragment = 1;
    End_SubFragment = 1;
end
end
%Compact SubFragment Array
SubFragment_stat_prior(~any(SubFragment_stat_prior,2), :) = [];
SubFragment_stat_prior_tmp = zeros(1,3*No_of_labeled_points+4);
n=1;
for m=1:size(SubFragment_stat_prior,1)
    %remove subfragment that has Start and End Subfragment on same row
    if SubFragment_stat_prior(m,2) < SubFragment_stat_prior(m,3)
        SubFragment_stat_prior_tmp(n,:) = SubFragment_stat_prior(m,:);
        n=n+1;
    end
end
SubFragment_stat_prior = SubFragment_stat_prior_tmp;

%Consolidate SubFragments, such that subfragments separated less than
%the Max_SubFragment_Gap are considered as one subfragment
SubFragment_gap = zeros(100,1);
for m=1:99
    if ( m < size(SubFragment_stat_prior,1))
        SubFragment_gap(m) = SubFragment_stat_prior(m+1,2) - SubFragment_stat_prior(m,3);
        if SubFragment_gap(m) < Max_SubFragment_Gap
            SubFragment_stat_prior(m,3) = SubFragment_stat_prior(m+1,3);
            SubFragment_stat_prior(m+1, :) = [];
        end
    end
end
SubFragment_stat_latter = SubFragment_stat_prior;
for m=1:size(SubFragment_stat_prior,1)
    SubFragment_stat(m,1) = SubFragment_stat_prior(m,1)+Start_fragment;
    SubFragment_stat(m,2) = SubFragment_stat_prior(m,2)+Start_fragment;
    SubFragment_stat(m,3) = SubFragment_stat_prior(m,3)+Start_fragment;
end
SubFragment_stat(~any(SubFragment_stat,2), :) = [];
No_of_SubFragments = size(SubFragment_stat,1);
SubFragment_aux = zeros(No_of_SubFragments,17);

%Step 3)For each sub-fragment, average change in fingertips shall be
% smaller than average change in palm coordinates during
% wall-rearing; as fingertips movement are constrained while palm are
% free
%
for p=1:No_of_SubFragments

```

```

Start_SubFragment = SubFragment_stat_prior(p,2)+Start_fragment;
End_SubFragment = SubFragment_stat_prior(p,3)+Start_fragment;

Start_prior = Start_SubFragment - Inspection_window;
End_latter = End_SubFragment + Inspection_window;
Duration_inspection_window = End_latter - Start_prior + 1;
if Start_prior < 1
    Start_prior = 1;
end
if End_latter > no_of_rows
    End_latter = no_of_rows;
end

%Compute Statistics prior to congruence point
Sum_stable_1st_radius = 0;
Sum_stable_2nd_radius = 0;
Sum_stable_3rd_radius = 0;
Sum_stable_palm = 0;

Sum_stable_1st_radius_delta = 0;
Sum_stable_2nd_radius_delta = 0;
Sum_stable_3rd_radius_delta = 0;
Sum_stable_palm_delta = 0;

for q = Start_prior:Start_SubFragment
    %Compute average radial distance prior to Mid-subfragment
    Sum_stable_1st_radius = Sum_stable_1st_radius + Average_Polar_Radius(q,stable_1st);
    Sum_stable_2nd_radius = Sum_stable_2nd_radius + Average_Polar_Radius(q,stable_2nd);
    Sum_stable_3rd_radius = Sum_stable_3rd_radius + Average_Polar_Radius(q,stable_3rd);
    Sum_stable_palm = Sum_stable_palm + Average_Polar_Radius(q,No_of_labeled_points);

    %Compute total radial distance change prior to Start-SubFragment
    Sum_stable_1st_radius_delta = Sum_stable_1st_radius_delta + Average_Polar_Radius(q, ✓
stable_1st+No_of_labeled_points);
    Sum_stable_2nd_radius_delta = Sum_stable_2nd_radius_delta + Average_Polar_Radius(q, ✓
stable_2nd+No_of_labeled_points);
    Sum_stable_3rd_radius_delta = Sum_stable_3rd_radius_delta + Average_Polar_Radius(q, ✓
stable_3rd+No_of_labeled_points);
    Sum_stable_palm_delta = Sum_stable_palm_delta + Average_Polar_Radius(q, ✓
2*No_of_labeled_points);

end

Duration_of_SubFragment = Start_SubFragment - Start_prior;
if (Duration_of_SubFragment > 0)
    %Average Radial distances of FingerTips & Palm
    SubFragment_stat_prior(p, 3+stable_1st) = Sum_stable_1st_radius / double ✓
(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+stable_2nd) = Sum_stable_2nd_radius / double ✓
(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+stable_3rd) = Sum_stable_3rd_radius / double ✓

```



```

(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+No_of_labeled_points) = Sum_stable_palm / double ✓
(Duration_of_SubFragment);
    %Average change in FingerTips & Palm per frame
    SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_1st) = ✓
Sum_stable_1st_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_2nd) = ✓
Sum_stable_2nd_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_3rd) = ✓
Sum_stable_3rd_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_prior(p, 3+3*No_of_labeled_points) = Sum_stable_palm_delta / double ✓
(Duration_of_SubFragment);
    else
        SubFragment_stat_prior(p,:) = 0;
    end

    %Compute Statistics after congruence point
    Sum_stable_1st_radius = 0;
    Sum_stable_2nd_radius = 0;
    Sum_stable_3rd_radius = 0;
    Sum_stable_palm = 0;

    Sum_stable_1st_radius_delta = 0;
    Sum_stable_2nd_radius_delta = 0;
    Sum_stable_3rd_radius_delta = 0;
    Sum_stable_palm_delta = 0;

    for q = End_SubFragment:End_latter
        %Compute average radial distance prior to Mid-subfragment
        Sum_stable_1st_radius = Sum_stable_1st_radius + Average_Polar_Radius(q,stable_1st);
        Sum_stable_2nd_radius = Sum_stable_2nd_radius + Average_Polar_Radius(q,stable_2nd);
        Sum_stable_3rd_radius = Sum_stable_3rd_radius + Average_Polar_Radius(q,stable_3rd);
        Sum_stable_palm = Sum_stable_palm + Average_Polar_Radius(q,No_of_labeled_points);

        %Compute total radial distance change prior to Mid-SubFragment
        Sum_stable_1st_radius_delta = Sum_stable_1st_radius_delta + Average_Polar_Radius(q, ✓
stable_1st+No_of_labeled_points);
        Sum_stable_2nd_radius_delta = Sum_stable_2nd_radius_delta + Average_Polar_Radius(q, ✓
stable_2nd+No_of_labeled_points);
        Sum_stable_3rd_radius_delta = Sum_stable_3rd_radius_delta + Average_Polar_Radius(q, ✓
stable_3rd+No_of_labeled_points);
        Sum_stable_palm_delta = Sum_stable_palm_delta + Average_Polar_Radius(q, ✓
2*No_of_labeled_points);

    end

    Duration_of_SubFragment = End_latter - End_SubFragment;
    if (Duration_of_SubFragment > 0)
        %Average Radial distances of FingerTips & Palm
        SubFragment_stat_latter(p, 3+stable_1st) = Sum_stable_1st_radius / double ✓
(Duration_of_SubFragment);
        SubFragment_stat_latter(p, 3+stable_2nd) = Sum_stable_2nd_radius / double ✓

```

```

(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+stable_3rd) = Sum_stable_3rd_radius / double ✓
(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+No_of_labeled_points) = Sum_stable_palm / double ✓
(Duration_of_SubFragment);
    %Average change in FingerTips & Palm per frame
    SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_1st) = ✓
Sum_stable_1st_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_2nd) = ✓
Sum_stable_2nd_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_3rd) = ✓
Sum_stable_3rd_radius_delta / double(Duration_of_SubFragment);
    SubFragment_stat_latter(p, 3+3*No_of_labeled_points) = Sum_stable_palm_delta / double ✓
(Duration_of_SubFragment);
    else
        SubFragment_stat_latter(p,:) = 0;
    end

%**** Compute statistics for entire SubFragment ****%
Radii_1 = zeros(End_latter - Start_prior + 1,1);
Radii_2 = zeros(End_latter - Start_prior + 1,1);
Radii_3 = zeros(End_latter - Start_prior + 1,1);
Palm_1 = zeros(End_latter - Start_prior + 1,1);

for q = Start_prior:End_latter
    %Compute average radial distance prior to Mid-subfragment
    Radii_1(q - Start_prior + 1) = Average_Polar_Radius(q,stable_1st);
    Radii_2(q - Start_prior + 1) = Average_Polar_Radius(q,stable_2nd);
    Radii_3(q - Start_prior + 1) = Average_Polar_Radius(q,stable_3rd);
    Palm_1(q - Start_prior + 1) = Average_Polar_Radius(q,No_of_labeled_points);
end

SubFragment_aux(p,1) = Start_SubFragment;
SubFragment_aux(p,2) = End_SubFragment;
SubFragment_aux(p,2+stable_1st) = mean(Radii_1);
SubFragment_aux(p,2+stable_2nd) = mean(Radii_2);
SubFragment_aux(p,2+stable_3rd) = mean(Radii_3);
SubFragment_aux(p,2+No_of_labeled_points) = mean(Palm_1);
SubFragment_aux(p,10+stable_1st) = std(Radii_1);
SubFragment_aux(p,10+stable_2nd) = std(Radii_2);
SubFragment_aux(p,10+stable_3rd) = std(Radii_3);
SubFragment_aux(p,10+No_of_labeled_points) = std(Palm_1);

outward = 0;
if SubFragment_aux(p,2+stable_1st) > SubFragment_aux(p,2+No_of_labeled_points)
    outward = outward + 1;
end
if SubFragment_aux(p,2+stable_2nd) > SubFragment_aux(p,2+No_of_labeled_points)
    outward = outward + 1;
end
if SubFragment_aux(p,2+stable_3rd) > SubFragment_aux(p,2+No_of_labeled_points)
    outward = outward + 1;
end

```



```

end
%Compute Fingertips in outward orientation
SubFragment_aux(p,3+No_of_labeled_points)=outward;

below_inner = 0;
if SubFragment_aux(p,2+stable_1st) <= Inner_Diameter * safety_margin / 2
    below_inner = below_inner + 1;
end
if SubFragment_aux(p,2+stable_2nd) <= Inner_Diameter * safety_margin / 2
    below_inner = below_inner + 1;
end
if SubFragment_aux(p,2+stable_3rd) <= Inner_Diameter * safety_margin / 2
    below_inner = below_inner + 1;
end
%Compute Fingertips within inner diameter
SubFragment_aux(p,4+No_of_labeled_points)=below_inner;

SubFragment_aux(p,17) = (SubFragment_aux(p,2+No_of_labeled_points)+SubFragment_aux(p, ✓
2+stable_1st)+SubFragment_aux(p,2+stable_2nd)+SubFragment_aux(p,2+stable_3rd))/4;

%Discern fingertips and palm movement dynamics within
%congruent subfragment ie +/- 0.5 seconds from coherent minima
stable_1st_mvt = mvtSymbol(SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_1st), ✓
SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_1st));
stable_2nd_mvt = mvtSymbol(SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_2nd), ✓
SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_2nd));
stable_3rd_mvt = mvtSymbol(SubFragment_stat_prior(p, 3+2*No_of_labeled_points + stable_3rd), ✓
SubFragment_stat_latter(p, 3+2*No_of_labeled_points + stable_3rd));
stable_palm_mvt = mvtSymbol(SubFragment_stat_prior(p, 3+3*No_of_labeled_points), ✓
SubFragment_stat_latter(p, 3+3*No_of_labeled_points));

SubFragment_stat(p,2)=Start_SubFragment;
SubFragment_stat(p,3+stable_1st) = stable_1st_mvt;
SubFragment_stat(p,3+stable_2nd) =stable_2nd_mvt;
SubFragment_stat(p,3+stable_3rd) =stable_3rd_mvt;
SubFragment_stat(p,9) =stable_palm_mvt;

SubFragment_stat(p,10) = touchDetect(Start_SubFragment, End_SubFragment, stable_1st_mvt, ✓
stable_2nd_mvt, stable_3rd_mvt, stable_palm_mvt);

end

Fragment_header = zeros(1,Columns_of_SubFragments);
Fragment_header(1,1) = Start_fragment;
Fragment_header(1,2) = End_fragment;
Fragment_data_warehouse = cat(1,Fragment_data_warehouse,Fragment_header);
Fragment_data_warehouse = cat(1,Fragment_data_warehouse,SubFragment_stat);

Fragment_aux_header = zeros(1,17);
Fragment_aux_header(1,1) = Start_fragment;
Fragment_aux_header(1,2) = End_fragment;
Fragment_data_aux = cat(1,Fragment_data_aux,Fragment_aux_header);

```

```

Fragment_data_aux = cat(1,Fragment_data_aux,SubFragment_aux);

end
Fragment_data_warehouse(1,:) = [] ;
Fragment_data_aux(1,:) = [] ;

%***** Extract Congruent Points Specifications *****
%Initialize wall rearing fragment table
Fragment_congruence = zeros(1,10);

for k=1:No_of_fragments

    %For each fragment, inspect existence of the following evidence to
    %confirm of wall rearing
    %
    %Three possible situations will happen as paw hits the wall, be obstructed
    %and bounced back
    %
    %1a)Distance between finger tips and palm
    % shall demonstrate congruent minima for some fingers while decreasing
    % distance for others; it will not be surprising, due to acrylic
    % reflection, that some of these distances are negative, as DLC
    % recognize mirror reflection of palm based on its virtual image.
    %
    %1b)Change in radial distances of fingertips and palm from the center
    % shall be peaked (and possibly reversed)
    %
    %1c)Fingertips may linger on the wall (ie change of fingertips radial
    % distance will be hovering < 1 pics per frame

    %Acquire start and end of fragment, and compute its duration

    Start_fragment = Area_statistics(k,1);
    End_fragment = Area_statistics(k,2);
    Duration_of_fragment = End_fragment - Start_fragment +1;

    %
    %1) Identify the top three fingertips that demonstrate stability, and
    % record its start and end sub_fragment timings;

    %2) Identify the stable Palm position within the sub_fragment timing
    % where consecutive radial distance are less than Max_Palm_Threshold;
    %
    %3) Compute the average radial distance of the finger tips and the palm;
    %
    %4) Check if Palm radial distance is less than finger tips; if yes, paw
    % is oriented outwards, else it is oriented inwards

    paw = zeros(No_of_labeled_points,2);
    paw(1,2) = 1; % 1 = Thumb finger
    paw(2,2) = 2; % 2 = index finger
    paw(3,2) = 3; % 3 = Middle finger

```

```

paw(4,2) = 4; % 4 = Ring finger
paw(5,2) = 5; % 5 = small finger
paw(6,2) = 6; % 6 = palm
for i=Start_fragment:End_fragment
    %Compute for individual finger how many frames demonstrate slow
    %movements ie ?(radial distance) < Max_Palm_Threshold
    for j = 2*No_of_labeled_points+1:2*No_of_labeled_points+5
        if Fragment_Polar_Radius(i,j) > 0
            switch j
                case 2*No_of_labeled_points+1
                    paw(1,1) = paw(1,1) + 1;
                case 2*No_of_labeled_points+2
                    paw(2,1) = paw(2,1) + 1;
                case 2*No_of_labeled_points+3
                    paw(3,1) = paw(3,1) + 1;
                case 2*No_of_labeled_points+4
                    paw(4,1) = paw(4,1) + 1;
                otherwise
                    paw(5,1) = paw(5,1) + 1;
            end
        end
    end
    %Identify the top 3 fingertips that are most stable within the
    %fragment
    sort_paw = sortrows(paw, 'descend');
    stable_1st = int8(sort_paw(1,2));
    stable_2nd = int8(sort_paw(2,2));
    stable_3rd = int8(sort_paw(3,2));
end

Fragment_Polar_Radius_tmp = zeros(Duration_of_fragment, No_of_labeled_points * 4);
for i=Start_fragment:End_fragment
    for j = 1:No_of_labeled_points
        switch j
            case stable_1st
                Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
            case stable_2nd
                Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
            case stable_3rd
                Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
                Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
        end
    end
end

```

```

        Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,No_of_labeled_points+j);
        Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,2*No_of_labeled_points+j);
        case No_of_labeled_points
            Fragment_Polar_Radius_tmp(i-Start_fragment+1, j)=Average_Polar_Radius_temp3(i,j);
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,j);
            Fragment_Polar_Radius_tmp(i-Start_fragment+1,2*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,No_of_labeled_points+j);
            %Fragment_Polar_Radius_tmp(i-Start_fragment+1,3*No_of_labeled_points+j) = ✓
Fragment_Polar_Radius(i,2*No_of_labeled_points+j);
        end
    end
end

%1) Identify sub-fragments within the fragment where palm is shortest
% distance (ie Congruent Minima) from the stable fingertips; in these
% settings the paw is either grabbing something
% or pushing against the wall
%
%2) Back-trace the minima to the last moving palm to establish
% sub-fragment
%
%3) For each sub-fragment, average change in fingertips shall be
% smaller than average change in palm coordinates during
% wall-rearing; as fingertips movement are constrained while palm are
% free
%
for m=1:Duration_of_fragment
    %Flag stable palm sub-fragment
    if abs(Fragment_Polar_Radius_tmp(m,No_of_labeled_points*3)) <= Max_Palm_Threshold
        % ?(Palm) < Max_Palm_Threshold => Flag Stable Palm
        Fragment_Polar_Radius_tmp(m,No_of_labeled_points*4) = 1;
    else
        Fragment_Polar_Radius_tmp(m,No_of_labeled_points*4) = 0;
    end
end

%Extract minima fingertips - Palm distances
Stable_1st_finger = zeros(Duration_of_fragment, 1);
Stable_2nd_finger = zeros(Duration_of_fragment, 1);
Stable_3rd_finger = zeros(Duration_of_fragment, 1);
for m=1:Duration_of_fragment
    Stable_1st_finger(m,1) = -Fragment_Polar_Radius_tmp(m,No_of_labeled_points+stable_1st);
    Stable_2nd_finger(m,1) = -Fragment_Polar_Radius_tmp(m,No_of_labeled_points+stable_2nd);
    Stable_3rd_finger(m,1) = -Fragment_Polar_Radius_tmp(m,No_of_labeled_points+stable_3rd);

    [Min_1st, Min_1st_idx] = findpeaks(Stable_1st_finger);
    [Min_2nd, Min_2nd_idx] = findpeaks(Stable_2nd_finger);
    [Min_3rd, Min_3rd_idx] = findpeaks(Stable_3rd_finger);
end

```



```

end

%Group the trough points into clusters into sub_fragments
Congruent_Minima_index = zeros(1000,3);
Congruent_Mode_index = zeros(1000,1);
for l=1:size(Min_1st_idx)
    Congruent_Minima_index(int16(Min_1st_idx(l)),1)=1;
end
for l=1:size(Min_2nd_idx)
    Congruent_Minima_index(int16(Min_2nd_idx(l)),2)=1;
end
for l=1:size(Min_3rd_idx)
    Congruent_Minima_index(int16(Min_3rd_idx(l)),3)=1;
end
for p=1:1000-Congruence_window
    %where minima is observed, inspect next neighbor frames where
    %minima will also be observed
    if Congruent_Minima_index(p,1) > 0 || Congruent_Minima_index(p,2) > 0 || Congruent_Minima_index
    (p,3) > 0
        for q=0:Congruence_window
            Congruent_Mode_index (p) = Congruent_Mode_index (p) + Congruent_Minima_index(p+q,1)+
Congruent_Minima_index (p+q,2) + Congruent_Minima_index(p+q,3);
        end
    end
end
%Find congruence points where minima of fingertips - palm distances
%occur within 1 second among each other
[Congr_peaks_tmp, Congr_peaks_loc_tmp] = findpeaks(Congruent_Mode_index, 'MinPeakDistance',
Congruence_window);

r = 1;
Congr_peaks_loc = zeros(100,1);
Congr_peaks_loc_tmpl = Congr_peaks_loc_tmp;
for p=1:size(Congr_peaks_loc_tmp)
    if Congr_peaks_tmp(p) >= Min_Congruence
        %Fine tune location of Congruence Peaks
        %By extracting median within the Congruence_window
        Start_Congruence = int16(Congr_peaks_loc_tmp(p));
        Congr_median_array = zeros(3*(Congruence_window),1);
        for u=0:Congruence_window - 1
            for v = 1:3
                if Congruent_Minima_index(Start_Congruence + u, v) > 0
                    Congr_median_array(3*u+v) = Start_Congruence + u;
                end
            end
        end
        Congr_median_array = Congr_median_array(Congr_median_array ~= 0);
        Congr_peaks_loc_tmpl(p) = median(Congr_median_array);

        Congr_peaks_loc(r) = Congr_peaks_loc_tmpl(p);
        r=r+1;
    end
end

```

```

end
Congr_peaks_loc = Congr_peaks_loc(Congr_peaks_loc ~= 0);

No_of_SubFragments = size(Congr_peaks_loc,1);
SubFragment_stat_prior = zeros(No_of_SubFragments, 3*No_of_labeled_points+1);
SubFragment_stat_latter = zeros(No_of_SubFragments, 3*No_of_labeled_points+1);
Start_SubFragment = 1;
Mid_SubFragment = 1;
End_SubFragment = 1;
SubFragment_stat = zeros(No_of_SubFragments,10);
for p=1:No_of_SubFragments
    % Locate mid-points of maximum congruence between 3 stable
    % fingertips
    Mid_SubFragment = int16(Congr_peaks_loc(p))+Start_fragment;

    if Mid_SubFragment - Inspection_window > 1
        Start_SubFragment = Mid_SubFragment - Inspection_window;
    else
        Start_SubFragment = 1;
    end
    if Mid_SubFragment + Inspection_window < no_of_rows
        End_SubFragment = Mid_SubFragment + 12;
    else
        End_SubFragment = no_of_rows;
    end

    %*****Save Congruence Points *****

    SubFragment_stat(p,1) = Mid_SubFragment;
    SubFragment_stat(p,2) = Start_SubFragment;
    SubFragment_stat(p,3) = End_SubFragment;

end

Fragment_congruence = cat(1,Fragment_congruence,SubFragment_stat);

end
Fragment_congruence(1,:) = [] ;

%***** Decisions Evaluation Specifications *****
%**** 1st Phase: Correlate Coherent Subfragments with Congruence Points

%**** initial gauge of wall rearing

No_coherent_Subfragments = size(Fragment_data_warehouse,1);
No_congruent_points = size(Fragment_congruence,1);

%Fragment_data_warehouse should only flag subfragments that are BOTH
%Coherent (i.e. three fingers moving below 5 pixels per frame) AND
%Congruence within 1 second of Coherence (i.e. FingerTips - Palm distance
%of slow moving fingers exhibi minima)

```



```

Fragment_data_warehouse_tmp = zeros(No_coherent_Subfragments, Columns_of_SubFragments);
for i=1:No_coherent_Subfragments
    if Fragment_data_warehouse(i,3) > 0 && Fragment_data_warehouse(i,10) > 0
        %Process Coherent SubFragment records which was tagged as plausible
        %wall-rearing
        %
        %For each subfragment
        %if Coherent Subfragment +/- 1 second possess congruence points
        %decision(s) = 1;
        %else decision = 0
        %
        Fragment_data_warehouse_tmp(i,:)=Fragment_data_warehouse(i,:);
        Start_SubFragment = Fragment_data_warehouse(i,2)-Frame_Rate;
        End_SubFragment = Fragment_data_warehouse(i,3)+Frame_Rate;
        No_of_congruence = 0;
        for j=1:No_congruent_points
            %Find Number of Congruence points within each coherent
            %subfragment
            if Fragment_congruence(j,1)>=Start_SubFragment && Fragment_congruence(j,1) ≤
≤End_SubFragment
                No_of_congruence = No_of_congruence + 1;
            end
        end
        if No_of_congruence > 0
            Fragment_data_warehouse_tmp(i,10) = 1;
            Fragment_data_warehouse_tmp(i,11) = No_of_congruence;
        else
            Fragment_data_warehouse_tmp(i,10) = 0;
            Fragment_data_warehouse_tmp(i,11) = 0;
        end
    else
        %save header records
        Fragment_data_warehouse_tmp(i,:) = Fragment_data_warehouse(i,:);
    end
end

Fragment_data_warehouse = Fragment_data_warehouse_tmp;

%**** 2nd Phase: Eliminate Stationary Movements
%***** Filter Subfragments that are unlikely to be wall rearing
%1) If Subfragment's palm demonstrate standard deviations <
%Crawling_low_threshold
%
%2) If Subfragment duration is less than Minimum_touch_duration
%
Fragment_data_warehouse_tmp = zeros(No_coherent_Subfragments, Columns_of_SubFragments);
for i=1:No_coherent_Subfragments
    if Fragment_data_warehouse(i,3) > 0
        if Fragment_data_warehouse(i,10) > 0
            %transfer details records where wall-rearing is suspected
            Fragment_data_warehouse_tmp(i,:) = Fragment_data_warehouse(i,:);
        end
    end
end

```

```

stationary_count = 0;
for j=1:6
    if Fragment_data_aux(i,10+j) > 0
        if Fragment_data_aux(i,10+j) <= Crawling_low_threshold
            stationary_count = stationary_count + 1;
        end
    end
end
if stationary_count > 1
    Fragment_data_warehouse_tmp(i,10)=0;
    Fragment_data_warehouse_tmp(i,11)=0;
end

if (Fragment_data_warehouse(i,3) - Fragment_data_warehouse(i,2)) < Minimum_touch_duration
    Fragment_data_warehouse_tmp(i,10)=0;
    Fragment_data_warehouse_tmp(i,11)=0;
end

else
    %save details record with no wall-rearing detected
    Fragment_data_warehouse_tmp(i,:) = Fragment_data_warehouse(i,:);
end
else
    Fragment_ref = Fragment_ref + 1;
    %save header records
    Fragment_data_warehouse_tmp(i,:) = Fragment_data_warehouse(i,:);
end
end

%*** 3rd Phase: Evaluate Subfragment Statistics
%**** Program starts ****
Average_Polar_Radius_temp4 = zeros(no_of_rows, 12);
Average_Polar_Radius_temp5 = zeros(no_of_rows, 12);
Fragment_data_warehouse_tmp2 = zeros(1000,4);
Fragment_data_warehouse_tmp3 = zeros(1000,4);

Fragment_data_warehouse_tmp1 = Fragment_data_warehouse_tmp;
for i=1:No_coherent_Subfragments
    if Fragment_data_warehouse_tmp1(i,3) > 0
        if Fragment_data_warehouse_tmp1(i,10) > 0
            %Enlarge surveillance to +/- Inspection Windows of SubFragment
            Start_SubFragment = int16(Fragment_data_warehouse_tmp1(i,2) - Inspection_window);
            End_SubFragment = int16(Fragment_data_warehouse_tmp1(i,3) + Inspection_window);
            if Start_SubFragment < 1
                Start_SubFragment = 1;
            end
            if End_SubFragment >= no_of_rows
                End_SubFragment = no_of_rows - 1;
            end

            %Investigate on enlarged subfragment

```

```

        for k=Start_SubFragment:End_SubFragment
            for j = 1:6
                Average_Polar_Radius_temp4(k,j) = Raw_Polar_Radius(k,j);
                Average_Polar_Radius_temp4(k,j+No_of_labeled_points) = Raw_Polar_Radius(k+1,j)- ✓
Raw_Polar_Radius(k,j);
            end
        end
    end
end

Average_Polar_Radius_temp5 = Average_Polar_Radius_temp4;
for i=1:no_of_rows
    Transition_counter = 0;
    for j=No_of_labeled_points+1:2*No_of_labeled_points
        if abs(Average_Polar_Radius_temp4(i,j)) >= Transition_TH
            Transition_counter = Transition_counter + 1;
        end
    end
    if Transition_counter > 1
        %Number of fingertips that exceed Transition_TH >= 2
        for j=1:2*No_of_labeled_points
            Average_Polar_Radius_temp5(i,j) = 0;
        end
    end
end

%Repartition SubFragment
k=1;
Start_Subfragment = 1;
SubFragment_statistics_aux = zeros(1000,3+2*No_of_labeled_points);
for i=1:no_of_rows-1
    if(Average_Polar_Radius_temp5(i,1) < 1 && Average_Polar_Radius_temp5(i+1,1)>= 1)
        %transition to start of repartitioned Subfragment
        Start_Subfragment = i+1;
    elseif (Average_Polar_Radius_temp5(i,1) >= 1 && Average_Polar_Radius_temp5(i+1,1)< 1)
        %transition to end of repartitioned Subfragment
        End_Subfragment = i;

        SubFragment_statistics_aux(k,1) = Start_Subfragment;
        SubFragment_statistics_aux(k,2) = End_Subfragment;

        k = k + 1;
    elseif (Average_Polar_Radius_temp5(no_of_rows,1) >= 1 && i == no_of_rows-1)
        %End of array treatment
        End_Subfragment = no_of_rows;

        SubFragment_statistics_aux(k,1) = Start_Subfragment;
        SubFragment_statistics_aux(k,2) = End_Subfragment;
    end
end
SubFragment_statistics_aux(~any(SubFragment_statistics_aux,2), : ) = [];

```

```

%Eliminate any repartitioned SubFragment that has less than 2 frames
No_repart_subfrag = size(SubFragment_statistics_aux,1);
for k=1:No_repart_subfrag
    if (SubFragment_statistics_aux(k,2) - SubFragment_statistics_aux(k,1) < Min_stat_computation)
        %If duration of frame transitions less than 3 time frames,
        %eliminate subfragments from statistics computation
        Start_SubFragment = int16(SubFragment_statistics_aux(k,1));
        End_SubFragment = int16(SubFragment_statistics_aux(k,2));
        for i=Start_SubFragment:End_SubFragment
            for j=1:2*No_of_labeled_points
                Average_Polar_Radius_temp5(i,j) = 0;
            end
        end
        SubFragment_statistics_aux(k,1) = 0;
        SubFragment_statistics_aux(k,2) = 0;

    end
end
SubFragment_statistics_aux(~any(SubFragment_statistics_aux,2), : ) = [];

%Compute statistics for each repartitioned SubFragment
No_repart_subfrag = size(SubFragment_statistics_aux,1);
for k=1:No_repart_subfrag
    Start_SubFragment = int16(SubFragment_statistics_aux(k,1));
    End_SubFragment = int16(SubFragment_statistics_aux(k,2));
    SubFragment_duration = End_SubFragment - Start_SubFragment + 1;

    finger_1=zeros(SubFragment_duration,1);
    finger_2=zeros(SubFragment_duration,1);
    finger_3=zeros(SubFragment_duration,1);
    finger_4=zeros(SubFragment_duration,1);
    finger_5=zeros(SubFragment_duration,1);
    palm=zeros(SubFragment_duration,1);

    m=1;
    for i=Start_SubFragment:End_SubFragment
        finger_1(m) = Average_Polar_Radius_temp5(i,1);
        finger_2(m) = Average_Polar_Radius_temp5(i,2);
        finger_3(m) = Average_Polar_Radius_temp5(i,3);
        finger_4(m) = Average_Polar_Radius_temp5(i,4);
        finger_5(m) = Average_Polar_Radius_temp5(i,5);
        palm(m) = Average_Polar_Radius_temp5(i,6);
        m=m+1;
    end

    SubFragment_statistics_aux (k,3) = mean(finger_1);
    SubFragment_statistics_aux (k,4) = mean(finger_2);
    SubFragment_statistics_aux (k,5) = mean(finger_3);
    SubFragment_statistics_aux (k,6) = mean(finger_4);
    SubFragment_statistics_aux (k,7) = mean(finger_5);

```

```

SubFragment_statistics_aux (k,8) = mean(palm);

SubFragment_statistics_aux (k,3+No_of_labeled_points) = std2(finger_1);
SubFragment_statistics_aux (k,4+No_of_labeled_points) = std2(finger_2);
SubFragment_statistics_aux (k,5+No_of_labeled_points) = std2(finger_3);
SubFragment_statistics_aux (k,6+No_of_labeled_points) = std2(finger_4);
SubFragment_statistics_aux (k,7+No_of_labeled_points) = std2(finger_5);
SubFragment_statistics_aux (k,8+No_of_labeled_points) = std2(palm);

end

DigitPeaks = zeros(No_repart_subfrag,16);
DigitPeaksFlag = zeros(No_repart_subfrag,1);
for k=1:8
    DigitPeaks(:,k) = SubFragment_statistics_aux (:,k);
end
for k=3:7 %Find maxima of mean digit radii for each digits

    if DigitPeaks(1,k) > DigitPeaks(2,k)
        DigitPeaks(1,k+6) = 1;
    else
        DigitPeaks(1,k+6) = 0;
    end
    if DigitPeaks(No_repart_subfrag,k) > DigitPeaks(No_repart_subfrag-1,k)
        DigitPeaks(No_repart_subfrag,k+6) = 1;
    else
        DigitPeaks(No_repart_subfrag,k+6) = 0;
    end

    for l=2:(No_repart_subfrag-1)
        if DigitPeaks(1,k) > DigitPeaks(l-1,k) && DigitPeaks(1,k) > DigitPeaks(l+1,k)
            DigitPeaks(1,k+6) = 1;
        else
            DigitPeaks(1,k+6) = 0;
        end
    end
end
end
DigitPeaks(:,14) = DigitPeaks(:,9)+DigitPeaks(:,10)+DigitPeaks(:,11)+DigitPeaks(:,12)+DigitPeaks(:,13);
for m=1:No_repart_subfrag
    %Compute Convolution Column of DigitPeaks
    if m > 1 && m < No_repart_subfrag
        DigitPeaks(m,15) = DigitPeaks(m,14) + DigitPeaks(m+1,14) + DigitPeaks(m-1,14);
    elseif m == No_repart_subfrag
        DigitPeaks(m,15) = DigitPeaks(m,14) + DigitPeaks(m-1,14);
    elseif m == 1
        DigitPeaks(m,15) = DigitPeaks(m,14) + DigitPeaks(m+1,14);
    end
end
end
for m=1:No_repart_subfrag
    if m > 1 && m < No_repart_subfrag
        if DigitPeaks(m,15) > 4
            DigitPeaks(m,16) = 1;

```



```

        DigitPeaks(m+1, 16) = 1;
        DigitPeaks(m-1, 16) = 1;
    end
elseif m == 1
    if DigitPeaks(m,15) > 4
        DigitPeaks(m,16) = 1;
        DigitPeaks(m+1,16) = 1;
    end
elseif m == No_repart_subfrag
    if DigitPeaks(m,15) > 4
        DigitPeaks(m,16) = 1;
    end
end
end
DigitPeaksFlag = DigitPeaks(:,16);
SubFragment_statistics_aux_backup = zeros(No_repart_subfrag, 15);
for m=1:No_repart_subfrag
    if DigitPeaksFlag(m,1) > 0
        SubFragment_statistics_aux_backup(m,:) = SubFragment_statistics_aux(m,:);
    end
end
SubFragment_statistics_aux_backup(~any(SubFragment_statistics_aux_backup,2), : ) = [];
SubFragment_statistics_aux = SubFragment_statistics_aux_backup;

%For plausible wall rearing, some fingertips shall demonstrate slow
%variations as it rest on wall, while others shall demonstrate mobility
No_repart_subfrag = size(SubFragment_statistics_aux,1);
for k=1:No_repart_subfrag
    count = 0;
    for j=1:No_of_labeled_points-1
        if SubFragment_statistics_aux(k,2+No_of_labeled_points+j) <= Variation_TH
            count = count + 1;
        end
    end
    SubFragment_statistics_aux(k,15) = count;
end

%Eliminate SubFragments of false positives for the following conditions
%1) Number of slow moving fingertips between 2 to 4
%2) Eliminate slow moving and fast moving palm
%   Slow moving implies entire paw is resting
%   Fast moving does not make posture sense
%3) Each fingertip cannot distant from each other more than 100 pixels
SubFragment_statistics_aux_tmp = SubFragment_statistics_aux;
for k=1:No_repart_subfrag
    clear_subfrag = 0;
    if (SubFragment_statistics_aux(k,15) < 2 || SubFragment_statistics_aux(k,15) > 4)
        %For wall rearing, the five fingertips has either 2,3, or 4
        %slow moving
        clear_subfrag = 1;
    end
end

```



```

if int16(SubFragment_statistics_aux(k,15)) == 4
    %if four fingertips are slow moving, inspect whether the palm is
    %slow moving or ultra fast moving
    if SubFragment_statistics_aux(k,14) <= Variation_TH
        clear_subfrag = 1;
    elseif SubFragment_statistics_aux(k,14) >= Variation_High_TH
        clear_subfrag = 1;
    end
end

%Compute maxima and minima of slow moving fingertips
slow_radix_matrix = zeros(int16(SubFragment_statistics_aux(k,15)),1);
m=1;
for j=1:(No_of_labeled_points-1)
    if SubFragment_statistics_aux(k,2+No_of_labeled_points + j) <= Variation_TH
        slow_radix_matrix(m) = SubFragment_statistics_aux(k,2+j);
        m=m+1;
    end
end
if max(slow_radix_matrix) - min(slow_radix_matrix) >= Max_radix_differentials
    clear_subfrag = 1;
end

    %Identify the three slowest moving fingertips and remove those that lie
    %within Inner_Radius
    Variations = zeros(5,1);
    for j=9:13
        Variations(j-8,1) = SubFragment_statistics_aux(k,j);
    end
    Low_var = min(Variations,3);
    for j=3:7
        for m = 1:3
            if SubFragment_statistics_aux(k,j+No_of_labeled_points) == Low_var(m)
                if SubFragment_statistics_aux(k,j) < Inner_radius_outlier_removal_factor*Inner_Diameter ✓
/2
                    clear_subfrag = 1;
                end
            end
        end
    end
end

if clear_subfrag == 1
    for j=1:(3+2*No_of_labeled_points)
        SubFragment_statistics_aux_tmp(k,j) = 0;
    end
end
end

%Remove all zeros rows data( ~any(data,2), : ) = []; %rows
SubFragment_statistics_aux_tmp( ~any(SubFragment_statistics_aux_tmp ,2),:) = [];

%Reconstruct SubFragment report
n=1;

```

```

No_repart_subfrag = size(SubFragment_statistics_aux_tmp,1);
for m=1:No_of_fragments
    Start_fragment = int16(Area_statistics(m,1));
    End_fragment = int16(Area_statistics(m,2));

    Fragment_data_warehouse_tmp2(n,1) = Start_fragment;
    Fragment_data_warehouse_tmp2(n,2) = End_fragment;
    n=n+1;

    for k=1:No_repart_subfrag
        Start_SubFragment = SubFragment_statistics_aux_tmp(k,1);
        End_SubFragment = SubFragment_statistics_aux_tmp(k,2);

        if (Start_SubFragment >= Start_fragment && Start_SubFragment < End_fragment) && End_SubFragment
            <= (End_fragment + Frame_Rate)
                Fragment_data_warehouse_tmp2(n,1) = int16((Start_SubFragment + End_SubFragment)/2);
                Fragment_data_warehouse_tmp2(n,2) = Start_SubFragment;
                Fragment_data_warehouse_tmp2(n,3) = End_SubFragment;
                Fragment_data_warehouse_tmp2(n,4) = 1;
                n = n+1;
            end
        end
    end
end
%Remove all zeros rows data( ~any(data,2), : ) = []; %rows
Fragment_data_warehouse_tmp2( ~any(Fragment_data_warehouse_tmp2 ,2), :) = [];

%-----%
%For fast moving SubFragments, Eliminate SubFragments of false positives for the following conditions
%1) Number of slow moving fingertips less than 2
%2) Identify three fingertips of slowest variations
%3) All fingertips has to lie beyond Inner Diameter
SubFragment_statistics_aux_tmp1 = SubFragment_statistics_aux;
No_repart_subfrag = size(SubFragment_statistics_aux,1);
for k=1:No_repart_subfrag
    clear_subfrag = 0;
    if SubFragment_statistics_aux(k,15) > 1
        %Eliminate all SubFragments that has more than 1 stable fingertips
        clear_subfrag = 1;
    end

    if clear_subfrag == 1
        for j=1:(3+2*No_of_labeled_points)
            SubFragment_statistics_aux_tmp1(k,j) = 0;
        end
    end
end
end
%Remove all zeros rows data( ~any(data,2), : ) = []; %rows
SubFragment_statistics_aux_tmp1( ~any(SubFragment_statistics_aux_tmp1 ,2), :) = [];

No_repart_subfrag = size(SubFragment_statistics_aux_tmp1,1);
SubFragment_statistics_aux_tmp5 = SubFragment_statistics_aux_tmp1;
for k=1:No_repart_subfrag

```

```

clear_subfrag = 0;

%Remove SubFragments where more than 3 digits radii distance lie
%within Inner_Radius

No_of_inner_digits = 0;
for j=3:8
    if SubFragment_statistics_aux_tmp5(k,j) < Inner_radius_outlier_removal_factor*Inner_Diameter /2
        No_of_inner_digits = No_of_inner_digits + 1;
        SubFragment_statistics_aux_flag(k,1) = No_of_inner_digits;
    end
end
if No_of_inner_digits > 3
    clear_subfrag = 1;
end

%Compute maxima and minima of slow moving fingertips
slow_radii_matrix = zeros(int16(SubFragment_statistics_aux(k,15)),1);
m=1;
for j=1:(No_of_labeled_points-1)
    if SubFragment_statistics_aux(k,2+No_of_labeled_points + j) <= Variation_TH
        slow_radii_matrix(m) = SubFragment_statistics_aux(k,2+j);
        m=m+1;
    end
end
if max(slow_radii_matrix) - min(slow_radii_matrix) >= Max_radii_differentials
    clear_subfrag = 1;
end

if clear_subfrag == 1
    for j=1:(3+2*No_of_labeled_points)
        SubFragment_statistics_aux_tmp5(k,j) = 0;
    end
end

end

%Remove all zeros rows data( ~any(data,2), : ) = []; %rows
SubFragment_statistics_aux_tmp5( ~any(SubFragment_statistics_aux_tmp5 ,2), :) = [];
SubFragment_statistics_aux_tmp1 = SubFragment_statistics_aux_tmp5;

%Reconstruct SubFragment report
n=1;
No_repart_subfrag = size(SubFragment_statistics_aux_tmp1,1);
for m=1:No_of_fragments
    Start_fragment = int16(Area_statistics(m,1));
    End_fragment = int16(Area_statistics(m,2));

    Fragment_data_warehouse_tmp3(n,1) = Start_fragment;
    Fragment_data_warehouse_tmp3(n,2) = End_fragment;
    n=n+1;

    for k=1:No_repart_subfrag
        Start_SubFragment = SubFragment_statistics_aux_tmp1(k,1);

```

```

End_SubFragment = SubFragment_statistics_aux_tmp1(k,2);

    if (Start_SubFragment >= Start_fragment && Start_SubFragment < End_fragment) && End_SubFragment ✓
    <= (End_fragment + Frame_Rate)
        Fragment_data_warehouse_tmp3(n,1) = int16((Start_SubFragment + End_SubFragment)/2);
        Fragment_data_warehouse_tmp3(n,2) = Start_SubFragment;
        Fragment_data_warehouse_tmp3(n,3) = End_SubFragment;
        Fragment_data_warehouse_tmp3(n,4) = 1;
        n = n+1;
    end
end
end
%Remove all zeros rows data( ~any(data,2), : ) = []; %rows
Fragment_data_warehouse_tmp3( ~any(Fragment_data_warehouse_tmp3 ,2),:) = [];

%***** Consolidate subfragments into episodes *****
Fragment_data_warehouse_tmp4 = zeros(1000,4);

%Eliminate zero rows of SubFragment statistics aux, and concatenate the two
%SubFragment statistics into one array
SubFragment_statistics_aux_tmp2 = SubFragment_statistics_aux_tmp;
SubFragment_statistics_aux_tmp3 = SubFragment_statistics_aux_tmp1;

SubFragment_statistics_aux_tmp2( ~any(SubFragment_statistics_aux_tmp2 ,2),:) = [];
SubFragment_statistics_aux_tmp3( ~any(SubFragment_statistics_aux_tmp3 ,2),:) = [];
SubFragment_statistics_aux_tmp4 = cat(1,SubFragment_statistics_aux_tmp2, ✓
SubFragment_statistics_aux_tmp3);
SubFragment_statistics_aux_tmp4 = sort(SubFragment_statistics_aux_tmp4);

n=1;
No_repart_subfrag = size(SubFragment_statistics_aux_tmp4,1);
for m=1:No_of_fragments
    Start_fragment = int16(Area_statistics(m,1));
    End_fragment = int16(Area_statistics(m,2));

    Fragment_data_warehouse_tmp4(n,1) = Start_fragment;
    Fragment_data_warehouse_tmp4(n,2) = End_fragment;
    n=n+1;

    for k=1:No_repart_subfrag
        Start_SubFragment = SubFragment_statistics_aux_tmp4(k,1);
        End_SubFragment = SubFragment_statistics_aux_tmp4(k,2);

        if Start_SubFragment >= Start_fragment && End_SubFragment <= End_fragment
            Fragment_data_warehouse_tmp4(n,1) = int16((Start_SubFragment + End_SubFragment)/2);
            Fragment_data_warehouse_tmp4(n,2) = Start_SubFragment;
            Fragment_data_warehouse_tmp4(n,3) = End_SubFragment;
            Fragment_data_warehouse_tmp4(n,4) = 1;
            n = n+1;
        end
    end
end
end

```

```

end
Fragment_data_warehouse_tmp4( ~any(Fragment_data_warehouse_tmp4 ,2),:) = [];

%***** Wall Rearing SubFragment Consolidation *****
%Time difference within which subfragments consolidate
Consolidate_SubFrag_time = 3;
%Minimum Episode Time Parameter refers to the least possible episode
%considered for reporting
Min_Episode_Time = 0.25;

Fragment_data_warehouse_tmp4 = zeros(1000,4);

%Eliminate zero rows of SubFragment statistics aux, and concatenate the two
%SubFragment statistics into one array
SubFragment_statistics_aux_tmp2 = SubFragment_statistics_aux_tmp;
SubFragment_statistics_aux_tmp3 = SubFragment_statistics_aux_tmp1;

SubFragment_statistics_aux_tmp2( ~any(SubFragment_statistics_aux_tmp2 ,2),:) = [];
SubFragment_statistics_aux_tmp3( ~any(SubFragment_statistics_aux_tmp3 ,2),:) = [];
SubFragment_statistics_aux_tmp4 = cat(1,SubFragment_statistics_aux_tmp2,
SubFragment_statistics_aux_tmp3);
SubFragment_statistics_aux_tmp4 = sort(SubFragment_statistics_aux_tmp4);

n=1;
No_repart_subfrag = size(SubFragment_statistics_aux_tmp4,1);
for m=1:No_of_fragments
    Start_fragment = int16(Area_statistics(m,1));
    End_fragment = int16(Area_statistics(m,2));

    Fragment_data_warehouse_tmp4(n,1) = Start_fragment;
    Fragment_data_warehouse_tmp4(n,2) = End_fragment;
    n=n+1;

    for k=1:No_repart_subfrag
        Start_SubFragment = SubFragment_statistics_aux_tmp4(k,1);
        End_SubFragment = SubFragment_statistics_aux_tmp4(k,2);

        if Start_SubFragment >= Start_fragment && End_SubFragment <= End_fragment
            Fragment_data_warehouse_tmp4(n,1) = int16((Start_SubFragment + End_SubFragment)/2);
            Fragment_data_warehouse_tmp4(n,2) = Start_SubFragment;
            Fragment_data_warehouse_tmp4(n,3) = End_SubFragment;
            Fragment_data_warehouse_tmp4(n,4) = 1;
            n = n+1;
        end
    end
end
Fragment_data_warehouse_tmp4( ~any(Fragment_data_warehouse_tmp4 ,2),:) = [];

%***** Wall Rearing SubFragment Consolidation *****
%Consolidate SubFragments to Episodes, where successive subfragment are
%less than 2 seconds from each other
No_coherent_SubFragments = size(Fragment_data_warehouse_tmp4,1);

```



```

Fragment_data_warehouse_final = zeros(1,4);
SubFragment_data_warehouse = zeros(1,4);
SubFragment_data_warehouse_tmp = zeros(1,4);
Consol_SubFragments = zeros(No_coherent_SubFragments, 1);
Start_consolidation = 0;
for i=1:No_coherent_SubFragments
    if Fragment_data_warehouse_tmp4(i,3) > 0
        Start_consolidation = 1;
        Fragment_data_line(1,:) = Fragment_data_warehouse_tmp4(i,:);
        Fragment_data_line(1,4) = 1;
        SubFragment_data_warehouse = cat(1, SubFragment_data_warehouse, Fragment_data_line);
    else
        if Start_consolidation > 0
            SubFragment_data_warehouse( ~any(SubFragment_data_warehouse,2), : ) = [];
            Size_SubFragment_data_warehouse = size(SubFragment_data_warehouse, 1);
            for m=1:Size_SubFragment_data_warehouse
                if m == 1
                    Start_Subfragment = SubFragment_data_warehouse(m,2);
                end
                if m < Size_SubFragment_data_warehouse
                    if SubFragment_data_warehouse(m+1,2) - SubFragment_data_warehouse(m,3) <= 2 * ✓
                        End_Subfragment = SubFragment_data_warehouse(m+1,3);
                    else
                        End_Subfragment = SubFragment_data_warehouse(m,3);
                        Fragment_data_line(1,1) = int16((Start_Subfragment + End_Subfragment)/2);
                        Fragment_data_line(1,2) = Start_Subfragment;
                        Fragment_data_line(1,3) = End_Subfragment;
                        Fragment_data_line(1,4) = 1;

                        %Save if Subfragment duration > Minimum Episode
                        %duration
                        if (End_Subfragment - Start_Subfragment) > Min_Episode_Time * Frame_Rate
                            SubFragment_data_warehouse_tmp = cat(1, SubFragment_data_warehouse_tmp, ✓
Fragment_data_line);
                        end

                        Start_Subfragment = SubFragment_data_warehouse(m+1,2);
                        Start_consolidation = 0;
                    end
                elseif m == Size_SubFragment_data_warehouse
                    End_Subfragment = SubFragment_data_warehouse(m,3);
                    Fragment_data_line(1,1) = int16((Start_Subfragment + End_Subfragment)/2);
                    Fragment_data_line(1,2) = Start_Subfragment;
                    Fragment_data_line(1,3) = End_Subfragment;
                    Fragment_data_line(1,4) = 1;

                    %Save if Subfragment duration > Minimum Episode
                    %duration
                    if (End_Subfragment - Start_Subfragment) > Min_Episode_Time * Frame_Rate
                        SubFragment_data_warehouse_tmp = cat(1, SubFragment_data_warehouse_tmp, ✓
Fragment_data_line);

```



```

end
SubFragment_data_warehouse_tmp( ~any(SubFragment_data_warehouse_tmp,2), : ) = [];

Fragment_data_warehouse_final = cat(1, Fragment_data_warehouse_final, ✓
SubFragment_data_warehouse_tmp);

SubFragment_data_warehouse_tmp = zeros(1,4);
SubFragment_data_warehouse = zeros(1,4);

Start_consolidation = 0;
end
end
else
SubFragment_data_warehouse = zeros(1,4);
end
end

if i == No_coherent_SubFragments %End of Fragment_data_warehouse_tmp4
if Start_consolidation > 0
SubFragment_data_warehouse( ~any(SubFragment_data_warehouse,2), : ) = [];
Size_SubFragment_data_warehouse = size(SubFragment_data_warehouse, 1);
for m=1:Size_SubFragment_data_warehouse
if m == 1
Start_Subfragment = SubFragment_data_warehouse(m,2);
end
if m < Size_SubFragment_data_warehouse
if SubFragment_data_warehouse(m+1,2) - SubFragment_data_warehouse(m,3) <= 2 * ✓
Frame_Rate

End_Subfragment = SubFragment_data_warehouse(m+1,3);
else
End_Subfragment = SubFragment_data_warehouse(m,3);
Fragment_data_line(1,1) = int16((Start_Subfragment + End_Subfragment)/2);
Fragment_data_line(1,2) = Start_Subfragment;
Fragment_data_line(1,3) = End_Subfragment;
Fragment_data_line(1,4) = 1;

%Save if Subfragment duration > Minimum Episode
%duration
if (End_Subfragment - Start_Subfragment) > Min_Episode_Time * Frame_Rate
SubFragment_data_warehouse_tmp = cat(1, SubFragment_data_warehouse_tmp, ✓
Fragment_data_line);

end

Start_Subfragment = SubFragment_data_warehouse(m+1,2);
Start_consolidation = 0;
end
elseif m == Size_SubFragment_data_warehouse
End_Subfragment = SubFragment_data_warehouse(m,3);
Fragment_data_line(1,1) = int16((Start_Subfragment + End_Subfragment)/2);
Fragment_data_line(1,2) = Start_Subfragment;
Fragment_data_line(1,3) = End_Subfragment;

```



```

Fragment_data_warehouse_final_tmp(n,1) = Start_fragment;
Fragment_data_warehouse_final_tmp(n,2) = End_fragment;
n=n+1;

for k=1:No_repart_subfrag
    Start_SubFragment = Fragment_data_warehouse_final(k,2);
    End_SubFragment = Fragment_data_warehouse_final(k,3);

    if Start_SubFragment >= Start_fragment && End_SubFragment <= End_fragment
        Fragment_data_warehouse_final_tmp(n,1) = int16((Start_SubFragment + End_SubFragment)/2);
        Fragment_data_warehouse_final_tmp(n,2) = Start_SubFragment;
        Fragment_data_warehouse_final_tmp(n,3) = End_SubFragment;
        Fragment_data_warehouse_final_tmp(n,4) = 1;
        n = n+1;
    end
end
end

Fragment_data_warehouse_final_tmp( ~any(Fragment_data_warehouse_final_tmp ,2), :) = [];
Fragment_data_warehouse_final = Fragment_data_warehouse_final_tmp;

%***** Output Files Specifications *****
warning('off','MATLAB:xlswrite:AddSheet');

Fragment_data_warehouse_final_tmp = zeros(1000,5);
Fragment_data_warehouse_size = size(Fragment_data_warehouse_final,1);

Fragment_serial_number = 1;
for i=1:Fragment_data_warehouse_size
    if Fragment_data_warehouse_final(i,3) < 1
        Fragment_data_warehouse_final_tmp(i,1) = Fragment_serial_number;

        Fragment_data_warehouse_final_tmp(i,2) = Fragment_data_warehouse_final(i,1);
        Fragment_data_warehouse_final_tmp(i,3) = Fragment_data_warehouse_final(i,2);

        Fragment_serial_number = Fragment_serial_number + 1;
    else
        Fragment_data_warehouse_final_tmp(i,1) = 0;

        Fragment_data_warehouse_final_tmp(i,2) = Fragment_data_warehouse_final(i,1);
        Fragment_data_warehouse_final_tmp(i,3) = Fragment_data_warehouse_final(i,2);
        Fragment_data_warehouse_final_tmp(i,4) = Fragment_data_warehouse_final(i,3);
        Fragment_data_warehouse_final_tmp(i,5) = Fragment_data_warehouse_final(i,4);
    end
end
end
Fragment_data_warehouse_final_tmp( ~any(Fragment_data_warehouse_final_tmp,2), : ) = [];

report_header_1 = ["Fragment Number","Start Fragment","End Fragment"];
report_header_2 = ["","SubFragment Midpoint", "Start SubFragment", "End_SubFragment","Decision"];

```

```

if Process_left_paw == 1
    xlswrite(outputfile,report_header_1,"Left Paw Summary", "B1");
    xlswrite(outputfile,report_header_2,"Left Paw Summary", "B2");
    xlswrite(outputfile,Fragment_data_warehouse_final_tmp,"Left Paw Summary", "B3");

    xlswrite(outputfile,RawTable_final,"Left Paw Details", "A1");
else
    xlswrite(outputfile,report_header_1,"Right Paw Summary", "B1");
    xlswrite(outputfile,report_header_2,"Right Paw Summary", "B2");
    xlswrite(outputfile,Fragment_data_warehouse_final_tmp,"Right Paw Summary", "B3");

    xlswrite(outputfile,RawTable_final,"Right Paw Details", "A1");
end
clear;

end

%***** Functions *****

function mvt_symbol = mvtSymbol(prior,latter)

    if prior > 1
        if latter > 1
            mvt_symbol = 15;
        elseif 0 < latter && latter <=1
            mvt_symbol = 14;
        elseif -1 <= latter && latter <=0
            mvt_symbol = 12;
        else
            mvt_symbol = 13;
        end
    elseif 0 < prior && prior <= 1
        if latter > 1
            mvt_symbol = 11;
        elseif 0 < latter && latter <=1
            mvt_symbol = 10;
        elseif -1 <= latter && latter <=0
            mvt_symbol = 8;
        else
            mvt_symbol = 9;
        end
    elseif -1 <= prior && prior <=0
        if latter > 1
            mvt_symbol = 3;
        elseif 0 < latter && latter <=1
            mvt_symbol = 2;
        elseif -1 <= latter && latter <=0
            mvt_symbol = -1;
        else
            mvt_symbol = 1;
        end
    end
end

```

```

else
    if latter > 1
        mvt_symbol = 7;
    elseif 0 < latter && latter <=1
        mvt_symbol = 6;
    elseif -1 <= latter && latter <=0
        mvt_symbol = 4;
    else
        mvt_symbol = 5;
    end
end
end

function decision = touchDetect(start, finish, first, second, third, palm )

    palm_forward = 0;
    palm_rearing = 0;
    palm_fast_slow_forward = 0;
    palm_slow_slow_forward = 0;
    palm_fast_slow_retrace = 0;
    palm_slow_slow_retrace = 0;
    palm_slow_fast_forward = 0;

    first_forward = 0;
    first_rearing = 0;
    first_fast_slow_forward = 0;
    first_slow_slow_forward = 0;
    first_fast_slow_retrace = 0;
    first_slow_slow_retrace = 0;
    first_slow_fast_forward = 0;

    second_forward = 0;
    second_rearing = 0;
    second_fast_slow_forward = 0;
    second_slow_slow_forward = 0;
    second_fast_slow_retrace = 0;
    second_slow_slow_retrace = 0;
    second_slow_fast_forward = 0;

    third_forward = 0;
    third_rearing = 0;
    third_fast_slow_forward = 0;
    third_slow_slow_forward = 0;
    third_fast_slow_retrace = 0;
    third_slow_slow_retrace = 0;
    third_slow_fast_forward = 0;

    switch palm
        %Detect Palm rearing pattern before and after coherent subfragments
        case 15
            palm_forward = 1;
        case 14

```

```

        palm_fast_slow_forward = 1;
case 13
    palm_rearing = 1;
case 12
    palm_rearing = 1;
    palm_fast_slow_retrace = 1;
case 9
    palm_rearing = 1;
case 8
    palm_rearing = 1;
    palm_slow_slow_retrace = 1;
otherwise
    palm_forward = 0;
    palm_rearing = 0;
    palm_fast_slow_forward = 0;
    palm_fast_slow_retrace = 0;
    palm_slow_slow_retrace = 0;
end

switch first
case 15
    first_forward = 1;
case 14
    first_fast_slow_forward = 1;
case 13
    first_rearing = 1;
case 12
    first_rearing = 1;
    first_fast_slow_retrace = 1;
case 11
    first_slow_fast_forward = 1;
case 10
    first_slow_slow_forward = 1;
case 9
    first_rearing = 1;
    first_slow_fast_retrace = 1;
case 8
    first_rearing = 1;
    first_slow_slow_retrace = 1;
otherwise
    first_forward = 0;
    first_rearing = 0;
    first_fast_slow_forward = 0;
    first_slow_slow_forward = 0;
    first_fast_slow_retrace = 0;
    first_slow_slow_retrace = 0;
    first_slow_fast_forward = 0;
end

switch second
case 15

```



```

        second_forward = 1;
case 14
    second_fast_slow_forward = 1;
case 13
    second_rearing = 1;
case 12
    second_rearing = 1;
    second_fast_slow_retrace = 1;
case 11
    second_slow_fast_forward = 1;
case 10
    second_slow_slow_forward = 1;
case 9
    second_rearing = 1;
    second_slow_fast_retrace = 1;
case 8
    second_rearing = 1;
    second_slow_slow_retrace = 1;
otherwise
    second_forward = 0;
    second_rearing = 0;
    second_fast_slow_forward = 0;
    second_slow_slow_forward = 0;
    second_fast_slow_retrace = 0;
    second_slow_slow_retrace = 0;
    second_slow_fast_forward = 0;
end

```

```

switch third
case 15
    third_forward = 1;
case 14
    third_fast_slow_forward = 1;
case 13
    third_rearing = 1;
case 12
    third_rearing = 1;
    third_fast_slow_retrace = 1;
case 11
    third_slow_fast_forward = 1;
case 10
    third_slow_slow_forward = 1;
case 9
    third_rearing = 1;
    third_slow_fast_retrace = 1;
case 8
    third_rearing = 1;
    third_slow_slow_retrace = 1;
otherwise
    third_forward = 0;
    third_rearing = 0;

```

```

        third_fast_slow_forward = 0;
        third_slow_slow_forward = 0;
        third_fast_slow_retrace = 0;
        third_slow_slow_retrace = 0;
        third_slow_fast_forward = 0;
    end

    if (palm_rearing + first_rearing + second_rearing + third_rearing) >= 2
        %2 or more finger / palm demonstrate rearing pattern
        decision = 1;
    elseif (palm_rearing + first_rearing + second_rearing + third_rearing) > 0 && (palm_rearing +
first_rearing + second_rearing + third_rearing) < 2
        decision = 0.5;
    else
        decision = 0;
    end
end
end

```