Capturing Fast Motion with Consumer Grade Unsynchronized Rolling-Shutter Cameras

by

Xing Chen

B.Sc., Peking University, 2010

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

October 2012

Abstract

Motion capture plays an important role in a wide range of fields including film production, game industry, sports analysis and medicine. However, commercial optical motion capture systems are expensive due to their reliance on high speed high resolution cameras and accurate shutter synchronization. In the thesis, we present a technique based on continuous video exposure and stereo streak matching to do motion capture with a low cost, simple, portable and scalable setup of consumer grade cameras and LED markers. Neither shutter synchronization nor global shutters are required for geometric calibration and motion reconstruction. The system is presented in four stages, camera calibration and synchronization, streak segmentation and 2-D track extraction, stereo track matching and multi-view fusing. The system is qualitatively demonstrated through capturing human motion and fan spinning motion, and quantitatively evaluated by motions with known spatial trajectory and the rigidity of rigid moving objects.

Preface

The initial and high level ideas of the thesis project come from my supervisors, Prof. Robert Woodham and Prof. Wolfgang Heidrich. In particular, Prof. Woodham raised the idea of understanding motion from motion blur and Prof. Heidrich proposed to do motion capture with streaks.

I am the contributor of all the thesis work under the supervision of my supervisors.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to show my deep gratitude to my supervisors, Prof. Bob Woodham and Prof. Wolfgang Heidrich, for their valuable guidances, patience and supports throughout the thesis work. Particularly, I wish to thank Bob for his advice and amazing stories, which are far beyond the scholastic scope of the degree program but are extremely helpful to me. I also wish to thank Prof. Jim Little for his valuable advice as the second reader of the thesis.

I would also like to acknowledge my colleagues and friends who helped me in this thesis work. In particular, thank Wang Caoyu, Hu Maodi, Junaed Sattar, Fang Jing and Zhang Ling for helping me with the physical setups; thank Tang Yichen, Mattias Hullin, Gerwin Damberg, Ankur Gupta and Nasa Rouf for patiently answering my questions on the basics of electronics.

The thesis work benefits a lot from many open source projects including Python, NumPy, OpenCV, SciPy, Matplotlib, Mayavi, Cython and so on. I am very grateful to the awesome unselfish work of their developers.

Last but not least, I further extend my thanks to my friends and family members, here at UBC or far away on the other side of the ocean, for making me a happy life.

Chapter 1

Introduction

Recording motion has interested human beings for centuries because of its wide applications. For example, human motion capture has contributed to biomedical analysis, surveillance, sports performance analysis, human-computer interaction and film production [7, 11, 12, 22]. Besides, artists make animation based on motion observation [21]; biologists study animals through animal locomotion analysis [25]; in sports, knowing the motion of balls can also help making more convincing judgements during games and better visualization in broadcast video; in ballistics, much research and experiments are also based on motion analysis.

Specifically, recording motion or motion capture here refers to the procedure to record and reconstruct the 3-D positions of some interest points on objects over time. The other properties of motion, such as speed, rotational information or deformation, are not included in our definition of motion capture, but they can typically be inferred from the 3-D positions [7, 11].

A large variety of techniques, including but not restricted to optical, magnetic, inertial or mechanical ones, have been used for capturing motion of different subjects. Generally, non-optical systems tend to use complex sensors to record motion in a more direct way, while optical systems tend to use relatively basic sensors (mostly cameras) but more complicated computations. Optical systems can be further divided into two types: marker-based systems, where the actual subjects of motion capture are markers, and markerless systems, where computer vision techniques are applied to track motion of subjects directly [23].

A critical performance indicator for a motion capture system is its temporal resolution. Currently, most non-optical systems and low-end optical systems (such as Microsoft Kinect) are able to capture motion at 30-60Hz [7, 40], which is comparable to human vision perception rates. The rate can be enough for some applications like human-machine interaction, but for applications requiring more details of motion, it is generally far from enough due to the fast speed and complexity of some natural motion. Taking human motion as an example, a fastball throwing motion can easily be as fast as 140 km/h, which means that the human hand can travel more than 30 cm in 1/120 second; even during a casual jump, one can easily move 8 cm within 1/30 second [7].

Motion capture at high temporal and spatial resolution has mainly been done by advanced optical motion capture systems, which typically use high speed high resolution video cameras with short exposure time to get sharp images for stereo analysis. Although such systems have shown success in various applications, their implicit requirements of accurate synchronization make the setup very complex and are hardly portable or scalable in space. In addition, the huge cost of high speed high resolution camera arrays makes such systems prohibitive for potential users with limited budgets.

Through analysis, we believe the main reason for the reliance of high-end optical motion capture systems on those expensive high-end camera features and the complex setup is their requirements to get sharp images without motion blur. To do this for fast motion, a very short exposure is needed. Interestingly, although short exposure can enhance spatial accuracy at a specific moment, it also causes a severe loss of spatial-temporal information between exposures, which eventually leads to the requirements of high frame rate and accurate synchronization.

On the other hand, although motion blur is probably very bad for computer-vision based systems (which typically rely on image features), it may not be bad for marker-based systems where textures are not important. If one does not have to get rid of motion blur, a longer exposure can be used so that more information can then be recorded.

For video cameras, long exposure becomes continuous exposure (traditional long exposure for still cameras will not work for motion capture because little temporal information is encoded in a single photo). Here continuous exposure refers to keeping the shutter of a video camera open so that each frame of the recorded video is exposed in its whole frame duration. In such a mode, the camera will keep recording spatial information, and will also record some sparse temporal information. As long as one can effectively decode such information, a continuously exposed video can always provide more information than a video with short exposed sharp frames at the same frame rate. In other words, a comparable amount of information may be acquired from continuously exposed videos at a lower frame rate.

In this thesis, we show a system utilizing the information encoded in continuously exposed video to capture fast motion at a reasonably high spatial temporal resolution with relatively slow cameras. Further, as the proposed method does not require wired synchronization and can work with rolling shutter cameras, the whole setup is simple, portable and scalable, and can be built with only cheap devices which are available in consumer markets. As a demonstration, we show that consumer-level standard camcorders (1080/60i) are already able to produce decent results for human motion capture.

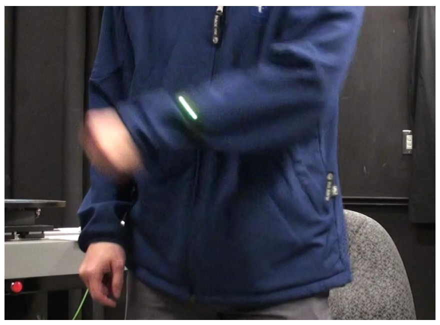
As the texture and shape of common objects will become hardly distinguishable under motion blur, we concentrate our efforts only on capturing the motion of texture-less bright point light sources. Motion capture of many other objects, such as human motion capture, is done by attaching small bright markers to the subject, as in other marker-based optical motion capture systems.

To be specific, under continuous exposure, a moving point light source will leave a streak on each camera, which is the projected spatial trajectory of the point light source. As long as the streaks are not self or mutually intersecting, reliable temporal information can be extracted from the endpoints of streaks. Also due to continuous exposure, in a video clip from one camera, streaks of the same point light source can be connected into a sparsely timestamped longer streak unless occlusion or detection failure occurs.

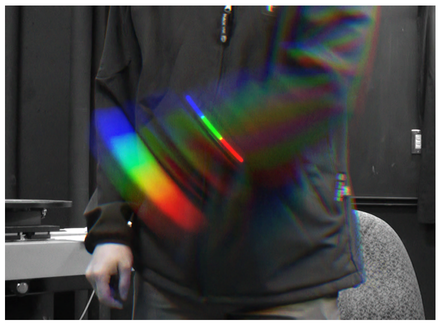
With a multi-camera setup, epipolar and temporal constraints can be used to match these streaks and 3-D spatial reconstruction can then be done. Due to the continuous exposure, accurate camera synchronization is not required. On the other hand, the temporal misalignments among cameras can actually be measured from stereo matching and can eventually increase the density of reliable timestamps on the reconstructed 3-D trajectory.

For clarity, in this thesis, the word "streaks" refers only to the thick curves on recorded video frames left by moving point light sources (see Fig. 1.1); "3-D tracks" or "3-D trajectories" both refer to paths of point light sources in the real world; the 2-D on-image projection of these trajectories is referred to as "2-D tracks" and "2-D trajectories", which are very likely the medial axis of the streaks. "Tracks" or "trajectories" can be associated with timestamps depending on the context.

Although the principle above is clear, many challenges exist for a robust implementation. On the cameras side, rolling shutter effects have to be dealt with carefully and camera calibration has to be done without wired synchronization; on image processing, robust streak segmentation has to be done on dimmed streaks with the presence of noise; on stereo matching, epipolar ambiguities, partial occlusions and broken or noisy trajectories can all bring challenges; finally, information obtained from different cameras has to be robustly fused. Besides, when the assumptions on streaks are not met (e.g. some streaks might intersect in some views), an elegant and minimal fail is desired.



(a) A blurred frame from a continuously exposed video. The streak is left by an LED.



(b) The intensity of three consecutive frames of the video are visualized in the red, green and blue channels of the image above. Note that due to continuous exposure, streaks from consecutive frames are perfectly connected.

Figure 1.1: A real example of the continuous exposure mode for a video camera and the streaks. The frames are extracted from a video recorded by SONY HDR-XR500 in 1080/60i mode. Note that continuous exposure can be easily done even if the environment is mildly bright.

In all, our contribution includes:

- A marker-streak based motion capture system based on cost effective consumer-level standard video cameras and a simple portable setup, which can be easily scaled in space.
- A simple way to calibrate an unsynchronized consumer video camera array (with or without rolling shutters) with a blinking LED.

The rest of the thesis is organized as follows: the second chapter is a review of related work; the third chapter gives a detailed discussion about our choices of camera modes and how synchronization and calibration is done on our low-end camera array; the fourth to the sixth chapters cover the details about motion reconstruction, including extracting tracks from images, matching tracks from different cameras and fusing information from different cameras, respectively; experiments evaluating the proposed system and demonstrations are presented in the seventh chapter; the final chapter is conclusion and discussion about future work. Some implementation details and basic math derivations are shown in the appendices.

Chapter 2

Related Work

Having been studied for centuries, recording motion can now be done with several quite different kinds of sensors, such as magnetic, inertial, mechanical and optical ones. As the underlying principle of non-optical sensors is completely different from optical ones and are not closely related our work, we only review optical motion capture techniques and systems, which mainly includes general high-speed photography and a specific motion capture topic, optical human motion capture.

Besides, because our system is built upon a low-end camera array, extra efforts have to be done to overcome the shortages of low-end cameras (e.g., rolling shutter effect). Background and research related to low-end cameras, camera synchronization and camera calibration are also reviewed.

2.1 High-Speed Photography

Photography is a way similar to human beings to perceive and record motion. Although only the 2-D projections of motion can be recorded directly through one camera, it is the foundation for most 3-D optical motion capture systems. The ideas of doing high speed photography with low-speed cameras are also inspiring for our work.

Early attempts of high speed photography date back to the 19-th century. In 1878, Eadweard Muybridge conducted a famous experiment and captured a sequence of photos of a galloping race horse. One of the photos, with the horse's four feet all off ground, answered a once famous debate about whether a horse always has at least one foot on ground even when galloping. The photos were captured by 12 stereoscopic cameras, each exposed once and in sequence [6, 25]. The setup is also an early prototype for achieving a high frame rate by multiple low frame rate cameras.

Later, during the decades when film cameras were dominating, several mechanical shutter based high-speed imaging techniques were invented. Intermittent motion, rotating prism and rotating mirror were three representative techniques during that period [17].

As photography entered the digital era, the introduction of electronic shutters has made high speed photography easier. Specifically, for CMOS sensors, one can set only a sub-array of a whole sensor as active, so that people are able to make trade offs between temporal and spatial resolution [27]. High speed cameras based on this mechanism are now easily accessible in the consumer market. As of August 2012, the $299.99 CASIO ZR100 camera is able to record low resolution (224×64) video at up to 1000 fps. However, for applications like motion capture, the limited spatial resolution provided by such fast cameras is still far from enough. On the other hand, high resolution high speed digital cameras always require a huge data bandwidth, which keeps such devices far beyond the scope of consumer market. As a result, efforts to do high speed photography using relatively slow cameras are still of interest.

In 2004, Wilburn et al. [38] developed the idea of increasing frame rate by using more cameras. A dense camera array of 52 CMOS cameras, each at 640×480 and 30 fps, were exposed sequentially in increments of $205\mu s$ and achieved 1560 fps at 640×480. However, important geometry information such disparities were ignored in their work and as a result, the virtual high speed camera will only work for 2-D or 3-D far-away scenes. Besides, the setup requires a very precise control on the exposure duration of cameras, which makes the whole setup complex and costly.

Instead of directly capturing high speed video at high resolution, some techniques record high speed motion without modifying cameras. For example, stroboscopic illumination was introduced to produce still or multi-exposed photographs of fast-moving objects. More recently, modern computer vision techniques were applied to strobe illuminated photos with multi-exposure to reconstruct fast motion. For example, in the work of Theobalt et al. [34], motion of a baseball and hands of baseball players were tracked; in the work of Linz et al. [20], dense motion vector fields were estimated. However, as information is lost between strobe flashes, such multi-exposure setups will not be able to work with arbitrary fast motions. Besides, the requirement that strobes must be able to significantly overpower ambient lighting limits the applications of strobes in bright environments.

Another technique to achieve high speed imaging is called “Streak Photography”, which increases camera speed by only sampling one dimension of spatial information at a time. The resultant photograph records time in one dimension and a single line of spatial information in the other. The technique was used in ballistic photography [10] and has lately been used to build a one dimensional camera at 0.4 trillion frames per second [36]. However, such cameras are always limited to one dimension and are not available to directly take photos of a 2-D scene. Notably, the word “streak” in “streak photography” is completely different from the word “streak” in this thesis, which refers to the 2-D projected spatial trajectory of a bright light source.

In all, despite the inspiring ideas that appeared during the exploration of high speed photography, no cheap way exists to record high speed video at a reasonably high resolution, which is required by most fast motion capture systems.

2.2 Optical Human Motion Capture

An optical motion capture system refers to a system which can reconstruct the 3-D positions of objects over time by processing data acquired from several optical image sensors. The other properties of the motion, such as speed and rotational information, are typically inferred from the 3-D positions [7, 11].

Human motion capture has become a particularly popular topic because of its wide applications and huge commercial potential in film production, video games, human-machine interaction, biomedical analysis and so on. Much research and commercial products of different spatial and temporal accuracy have appeared in the recent few years, which to some degree give a general profile of the problem of optical motion capture.

2.2.1 Marker Based Systems

The usage of markers is common in commercial motion capture systems to ease body part identification and to improve spatial accuracy. Markers are usually attached to human bodies and human motion is inferred by the reconstructed 3-D motion of markers.

Current commercial systems may either use passive markers (which reflect light) or active markers (which emit light by themselves). While the former has been widely used in many commercial systems [37] due to its simplicity, the latter is gaining more popularity in recent film productions because a unique identification of each marker can be achieved by modulating the active markers by time.

For applications requiring high temporal and spatial resolution (e.g., animating characters in films or video games), high speed cameras are used [28, 37]. The cameras have to be synchronized and of high spatial resolution to ensure 3-D triangulations can be done with high accuracy. As a result, such commercial motion capture systems are generally very expensive due to the high cost of the high speed imaging devices and can make it prohibitive for small productions.

2.2.2 Computer Vision Based Systems

Most of the recent research tends to try motion capture with computer vision techniques, so that human body parts can be identified automatically without any marker or manual labelling [24].

A well-known example is the Microsoft Kinect, which first does 2.5-D reconstruction of its view by analyzing projected infrared patterns and then identifies different parts of human body using depth information with machine learning based algorithms [31, 40]. As the system mainly targets low cost recreational purposes, it is available at relatively low cost and can be set up very easily. However, Kinect has a fairly low temporal resolution. Its frame rate of 30 Hz makes it unable to represent fast human motion exactly. Besides, the reliance on imaging projected light limits its operating space and makes large scale motion capture impossible.

Organic Motion, a computer vision based human motion capture system with a higher frame rate, is also available in the commercial market. With the help of 14 high speed FireWire synchronized cameras, the system is able to reconstruct 21-joint 3-D human motions without the help of markers at 60 Hz [28].

Generally, computer vision based human motion capture systems are friendly for the subject being captured as no special markers or sensors need to be worn. But the price of the high-end systems is still high due to the usage of expensive high speed cameras. Besides, computer vision based systems are typically designed for human motion capture only and can hardly be adapted to the motion capture of other objects. Even capturing scenes of people interacting with other objects is very challenging.

2.3 Motion Blur, Streaks and Long Exposure

Motion blur is the result of a relative motion between a camera and the scene during the relatively long exposure time of an image [26]. The typical outcome of a motion blur is the loss of high frequency information such as sharp image features in a photo. The motion blur of relatively small light sources under fast relative motion will cause streaks on the resulting photo.

Motion blur and streaks are annoying artifacts in many computer vision tasks mainly due to the loss of texture details where many image or video features are extracted from.

On the other hand, in the context of artistic photography, motion blur can be quite useful. For example, photographers may actively use neutral density filters to take pictures with longer exposure in a bright environment to get soft images; in night photography, a long exposure of a busy highway will result in beautiful streaks of the lights of vehicles; in light painting, trajectories of hand-held light sources can form a pretty drawing.

In scientific fields, motion blur has also shown its value. Chen et al. [5] and Fox et al. [9] estimated motion and depth from motion blur, respectively. Streaks can also be used to estimation camera motion for deblurring. However, the information one can get from a single blurred image is limited because of the lack of temporal information.

Scientific applications of trajectories are also found in physics. For example, the cloud chamber and the bubble chamber were invented to detect and record the path of electrically charged particles moving through them. Although their purposes and underlying theory are quite different from ours, the idea of capturing and analyzing fast motion by its trajectory is valuable.

The feature of continuous exposure on video cameras was mainly utilized in research based on fast strobes. For example, Bradley et al. [3] use strobes to synchronize cameras and to compensate for rolling shutter effects of consumer video cameras. To my knowledge research with continuously exposed video cameras under constant illumination has not been done before.

2.4 Rolling Shutter Cameras

Rolling shutter refers to the behaviour that different rows (scanlines) of a video frame are exposed in different time intervals. The behaviour is an outcome of the sequential-readout nature of camera sensor arrays built on complementary metal-oxide semiconductor (CMOS) technology, a technology which is cost effective and has become very popular in consumer-level cameras during the recent years [19].

The effect typically does not exist or has been minimized for machine vision cameras, but due to its popularity in the consumer market, much research has been done with rolling shutter cameras. For example, Liang et al. [19] analyzed the rolling shutter effect and proposed a compensation method; Wilburn et al. [38] made a fast camera array with rolling shutter cameras; Bradley et al. [3] compensated for rolling shutter by external strobes.

By now, the image formation model for cameras with rolling shutter has been well studied and can be used in our work.

2.5 Camera Array Synchronization

Cameras in a camera array generally need to be synchronized to utilize inter-camera information. There are two definitions of synchronization: one is to actively control the exposure so that cameras have same exposure durations, while the other is to passively measure the temporal offsets among cameras.

2.5.1 Active Exposure Control

Active exposure control is typically done via dedicated external wires. For example, Wilburn et al. [38] used CAT5 cables and a trigger signal generator to achieve an accuracy of $200ns$. However, as such high precision exposure control requires high-end cameras supporting hardware triggering and synchronization cables, the setup is generally costly and difficult to scale in space.

For cameras connected to and controlled by computers, less accurate camera synchronization (with a jitter of about 1-2ms) is reported to be achievable over network for Elphel cameras [8]. The method is simple and scalable, but still special devices are needed.

Instead of controlling cameras, an alternative way is to control environment illumination. Bradley et al. implemented virtual exposure control by strobes [3]. The method requires several video cameras in continuous exposure mode and a dark room where a strobe is the only light source, so that the actual exposure duration of the cameras is determined only by strobes. Synchronization is done by making the strobe blink in half of the cameras' frequency with a short "on" time in each cycle. The method can work with common low-end cameras and can even automatically remove rolling shutter effects. However, as strobe light has to be off for at least half of each frame's duration, the final output video of strobe-synchronization is restricted to limited exposure time, which implies a loss of information.

The idea of using blinking lights to synchronize cameras was found in the FoxTrax hockey puck tracking system back in 1997 [4]. In FoxTrax, the puck was equipped with several IR emitters pulsing with a 1/300 duty cycle. A sync detection system with 20 pulse detectors was built to detect the pulses and synchronize the cameras accordingly, so that the cameras would be able to see the pulse during their short exposure time. The system is of course too complex and expensive for general purposes.

2.5.2 Passive Temporal Offset Measurement

As all active exposure control methods we know do not work for our streak-based motion capture setup, we looked into the passive way, temporal offset measurement.

Generally, temporal offset measurement is done either by identifying events or by matching features by multi-view geometrical constraints. For example, in the work of Shrestha et al. [32], flashes from still cameras and sounds are used a cue for synchronization. In Lei and Yang's work [18], synchronization is done by shifting the timeline of tracked features so that the three-view geometry constraints are met. Depending on the specific method, temporal offsets measurement at frame or subframe accuracy can be done.

However, as far as we know, existing temporal offset measurement methods assume global shutters. Besides, as the task of streak-based motion capture itself can infer some temporal information, we decide to develop our own temporal offset measurement methods.

2.6 Camera Array Geometric Calibration

Multi-view stereo reconstruction in Euclidean spaces requires full geometric calibration of all cameras, including the intrinsic and extrinsic parameters.

2.6.1 Single Camera Calibration

Currently, popular single camera geometric calibration toolboxes (e.g., OpenCV and "Camera Calibration Toolbox for Matlab" [2]) are built based on Zhang's method [39], which is done by letting the cameras take images of a 2-D pattern with known geometry in different positions and orientations. Then, features on the images of the 2-D pattern are identified. Finally, camera parameters are estimated utilizing the rigidity and the known geometry of the 2-D pattern.

Popular 2-D patterns supported by current toolboxes include chessboard and circles grid. With either pattern, calibration can be done with (theoretically) at least two distinct views. However, to avoid being over-fitted to only the visible part of the view, around twenty views are practically needed for a calibration with good precision [1], which implies a considerable amount of human labour.

2.6.2 Multi-View Calibration

When inter-camera information is available, a more accurate and more robust calibration can generally be done. Such calibration methods are typically built on multi-view geometry [14] and Bundle Adjustment [35].

Extension to Zhang's Method

Zhang's method can of course be independently applied to each and every camera in a camera array, but it requires huge amounts of human labor. An alternative way is to present the 2-D pattern to all the cameras every time, so that one does not have to physically move the pattern too many times. However, as the position and orientation of cameras can be quite different, it is almost impossible to keep the whole pattern visible in all camera views while having a good coverage of the corners and borders of all views.

CALTag [1] by Atcheson et al. solved the problem above by introducing a new pattern, which is based on a chessboard but with distinct patterns embedded into every chessboard cell. With such pattern, a CALTag detector is able to robustly detect and identify a cropped or partially occluded CALTag pattern, so that one can freely move the chessboard in the space to be calibrated.

Some other research tries to improve calibration accuracy by utilizing inter-camera information (e.g., OpenCV's stereo calibration). This requires either statically placed calibration patterns or cameras with synchronized exposure. However, as placing calibration patterns is human labor intensive and synchronizing a rolling-shutter camera array is very difficult, the improvement is not practical in the context of this thesis.

The usage of blinking a LED in calibration was done by Koch et al. [16] for calibrating a distributed smart camera network. However, their purpose for using a blinking LED is to enhance the robustness of detection instead of synchronization or overcoming rolling-shutter effects, as done in this thesis.

2.6.3 Multi-Camera Self-Calibration

Svoboda et al. [33] proposed a different way to calibrate a synchronized camera array of more than three cameras with a single point light source, called "Multi-Camera Self-Calibration". To calibrate the cameras, one can just hold a point source (usually a laser pointer or an LED) by hand and move it freely in the space needed to be calibrated. As all the method needs is a single point light source, the method is scalable and simple to operate.

However, as the calibration is done by mainly utilizing inter-camera information, synchronization of the camera array is critical. As a result, the method cannot be directly applied to our camera array without synchronization (and possibly rolling shutter).

Chapter 3

Cameras

Like other optical motion capture systems, our streak based motion capture system needs several physically fixed video cameras to record motion. The choice of cameras and camera modes can make a big difference. In this chapter, we explain why inexpensive cameras without hardware synchronization can work for our method and discuss the system's requirements on cameras. Following that, the methods to do synchronization and geometrical calibration on an unsynchronized inexpensive camera array are also presented.

3.1 Motion Blur or Not?

Generally, when a dynamic scene is imaged by a camera, there is always a loss of motion information: for a moving object in general, it is not possible to simultaneously measure its position and motion tendency at a specific moment. The situation is the same for a video camera with a finite sampling rate. As a result, people have to make a trade-off between motion and position information.

In the context of capturing motion, if one wants to know the exact position of an object at sampled time points, the exposure time should be as short as possible to enhance temporal accuracy and avoid motion blur. On the other hand, if one cares more about the trajectory of a motion, a longer exposure is preferred.

3.1.1 Cost of Sharp Images

Current optical motion capture systems typically choose the first option and design their systems based on sharp video frames to minimize motion blur. The choice seems natural, but we believe it actually has a substantial cost, which can be the root cause for the complexity and expensiveness of current optical motion capture setups.

First, to deal with fast motion, the exposure time of each frame has to be extremely short to minimize motion blur. (Considering that a fastball pitching can reach 36 meters per second, an exposure time of shorter than

$1/1000s$ is definitely needed in such cases.) Such short exposure generally implies noisy frames due to higher sensor sensitivity and also implies that the shutters are closed for a significant portion of time, during which the system will be "blind" to whatever is happening in the world.

One direct outcome is that the trajectory of a motion can only be estimated from the sampled points instead of being measured directly. To get enough samples to well estimate a fast motion, such motion capture systems have to use high frame rate cameras, which are not easily available and generally quite expensive.

Moreover, as stereo matching and reconstruction algorithms have to be done on image contents recorded at the same moment, the short exposures of each camera have to be accurately synchronized. (For fastball pitching as an example again, a 10ms error in synchronization would cause a 36cm misalignment in space, which is very likely to break most stereo matching algorithms.) However, as high accuracy synchronization is typically done by dedicated sync cables, the wires inevitably make the setup complex and very hard to scale. Besides, the cameras have to be equipped with external trigger interfaces to support synchronization, which further increases the cost of the whole setup.

Another problem rises when multiple featureless objects need to be tracked, which is a common need in a marker based motion capture system: it is hard to know which marker in one frame corresponds to which marker in another frame. To solve this problem, some systems raise the speed of cameras so that a marker will not move too far in adjacent frames, while some others modulate the light emitted by markers in order to make each marker visually distinct. Both workarounds make the whole system even more complex and expensive.

The disadvantages lead us to reconsider the reasons to use motion blur free images. Some possible facts related to the choice are:

- Accurate temporal-spatial information at the sampled time points can be easily decoded from motion blur free images.
- Most motion irrelevant image features, such as the texture and shape of an object, only present in motion blur free images.

The second fact implies that computer vision based markerless systems, which need image features to do object recognition and stereo matching, can only work with motion blur free images. However, for marker based systems, as long as one can extract temporal-spatial information from blurred frames, motion reconstruction should be possible.

3.1.2 Motion from Streaks

To avoid the disadvantages of relying on sharp images and to do motion capture in the presence of motion blur, we propose to capture and reconstruct motion from streaks.

To be least affected by the loss of texture features during a motion blur, we design our systems based on plain constant point light sources. To maximize the information we can get about the trajectory of motion, we assume the video is in continuous exposure mode, in which mode the exposure time of each video frame is the same as the duration of the frame.

Under these settings, each moving light source will leave a streak on each camera frame (see Fig. 1.1). Each streak can convey important information, such as the trajectory of the light source's motion projected onto the camera view. The timestamp of the frame where a streak is from can also provide some temporal information.

Further, if the streaks are not self-intersecting and do not intersect with each other, more temporal information can be obtained. This is because streaks from consecutive frames in continuously exposed video should be touching. The touching point, which is exposed by the light source in both frames, can be labelled with an accurate timestamp, which is the read-out time for the first frame.

In addition, in a multi-view setup, accurate camera synchronization is not necessary any more because trajectories of the light sources are completely captured by the cameras. With the help of epipolar constraints, time stamps and basic physical principles about motion, points on streaks can be matched among cameras. Then 3-D trajectories can be reconstructed based on the matchings. In fact, if we are able to measure the temporal offset among cameras, an unsynchronized camera array can even help enhance the temporal resolution of the reconstructed trajectories because there will be more reliable timestamps on the trajectory.

Streaks also solve the correspondence problem by nature: streaks that are nicely connected between consecutive frames belong to the same object; streaks from different views which can be matched perfectly under temporal-geometrical constraints are also very likely from the same object.

Overall, the streak based system outlined above should be able to capture motion with an unsynchronized array of cameras at relatively lower frame rates, which are generally much cheaper and easier to set up than current commercial systems.

3.2 Requirements on Cameras

The core hardware of the streak based motion capture system is the cameras. Although we try to be as flexible as possible on the cameras, some basic requirements are still necessary.

3.2.1 Reliably Timestamped Frames

Timing is very important for motion. Timestamps, as the only temporal information source for captured motion, have to be reliable.

Luckily, almost all consumer-level cameras are able to conform to one of the popular video standards (such as NTSC and PAL), which require camcorders to produce videos at a fixed frame rate. For these cameras, the frame numbers of the output video can act as reliable timestamps.

A notable exception is that webcams, which are known to have varying frame rates, are typically unable to provide hardware-timestamped frames. Although it is possible to approximately attach timestamps to frames by software and save the video in a container video format with variable frame rate support (such as MKV), one should be aware of the potential error introduced by inaccurate timestamps if they capture motion with webcams.

3.2.2 Continuous Exposure

To get connectible streaks, the cameras need to be in continuous exposure mode. For video cameras, this means that the exposure time for each pixel on a frame should be equal to the frame duration. This is technically possible for digital camcorders because such cameras are typically equipped with a electronic shutter, so that the time for reading-out and reseting a sensor pixel can be negligible comparing to frame duration [19].

Practically, continuous exposure turns out to be the default setting for many consumer level camcorders and webcams when recording in a not too bright environment, such as a typical indoor scene or some dim outdoor scenes. Fig. 1.1 shows an example of three consecutive continuously exposed frames from a video clip.

3.2.3 Static Cameras

Camera motion is not considered in our motion capture system. All the cameras have to be static.

Practically, this can be done by using tripods or other camera mounts.

3.2.4 Number of Cameras

Our method is flexible with respect to the number of cameras. As most stereo reconstruction based systems, at least two fully calibrated cameras are required for our motion capture setup. However, a multi-view setup can generally be more robust and provide better results due to the following reasons:

- **Less Occlusion**. A setup with additional cameras generally has a better coverage of the space, so that a light source is more likely to be captured by multiple camera views.
- **Reduced Ambiguity**. As shown in Fig. 5.1, a two-view stereo setup is not able to well reconstruct motion within or near the epipolar plane. Such ambiguity can typically be resolved by bring in information from more views.
- **Consistency Check**. Two-view stereo may suffer from some matching errors, but the errors can rarely be consistent among three or more views. Validations based on multi-view consistency will make the algorithm much more robust to errors.
- **Easier Calibration**. As discussed in Section 3.5, calibration is easier to do with a larger camera array (at least three cameras are needed in our calibration method).

3.3 Notable Behaviours of Some Inexpensive Cameras

Our method makes it possible to capture motion with relatively slow cameras without a synchronization interface, which is a common configuration for cameras in the consumer market. However, to actually work with consumer level cameras, some other behaviours of these cameras should also be understood so that corrections can done accordingly. In this section, two notable behaviours of inexpensive cameras, interlacing and rolling shutter, are reviewed.

3.3.1 Interlacing

Some cameras record and produce video in interlaced mode, which means the even and odd scanlines of a frame are recorded in two separate fields.

The mode is known to enhance temporal resolution at the cost of losing some local spatial details [15].

For streak based motion capture tasks, we believe interlaced cameras are generally a better choice than progressive scanning cameras at the same frame rate because of the following reasons: (1) shorter exposure time and more accurate timestamps on a streak are preferred to avoid ambiguity and to improve accuracy; (2) local spatial details are relatively less important due to the continuous and smooth nature of streaks.

Practically, to adapt interlaced video to a normal frame based video processing framework, deinterlacing is done by turning fields into regular frames. The preferred deinterlacing algorithm for capturing motion is "Yadif (2x)" (the *Yet Another DeInterlacing Filter* from the MPlayer project), which reconstructs a full frame from every single field and produces seemingly progressive scanning videos at twice the original frame rate.

Besides, "continuous exposure" for interlacing cameras means continuous exposure for each field, or a virtual continuous exposure in deinterlaced video clips. In such mode, sensor pixels are actually exposed in only *half* of the time between two consecutive readouts. For example, for a $1080/60i$ camera, the exposure time of each sensor pixel is $1/60s$ in our continuous exposure mode, instead of $1/30s$. This may require an extra setting step on some cameras.

3.3.2 Rolling Shutter

Rolling shutter refers to the effect that different rows (scanlines) of one video frame are exposed in different time intervals.

To work with rolling shutter cameras, a comprehensive understanding of the shutter is important. From existing documents, we know that a time diagram for a typical low-cost CMOS sensor array can be drawn as Fig. 3.1. As shown in the figure, although the actual readout time for each scanline is very short, significant delay between consecutive readouts can occur for low-cost cameras because of the lack of frame buffer or a slow data transmission rate [19]. It has been shown that for many inexpensive CMOS cameras, the readout time for scanline y in frame j is [3]:

$$r_j^{(y)} = t_j + \frac{y}{S}\Delta t \tag{3.1}$$

Where t_j is the readout time for the first scanline in frame j and S is the total number of scanlines in a frame, including invisible scanlines.

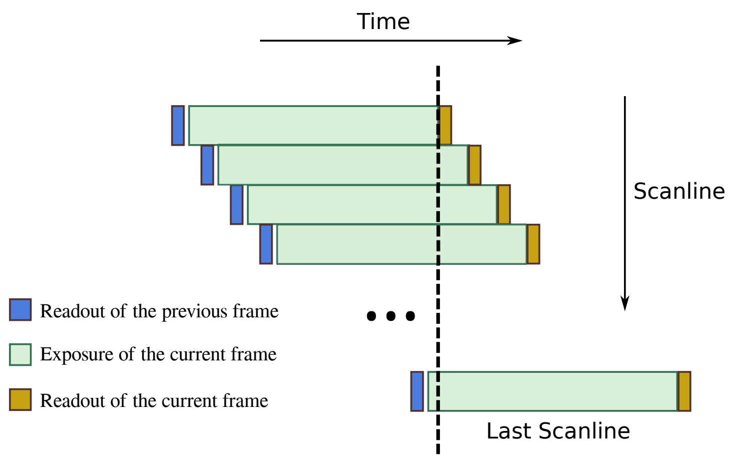

Time
Scanline
Readout of the previous frame
Exposure of the current frame
Readout of the current frame
Last Scanline

Figure 3.1: Timing diagram for a typical low-cost camera with a CMOS sensor array. Such a camera does not immediately read the next scanline when one readout is finished, but waits for a short while so that the time to read all the scanlines is roughly the same as the frame duration. (Note that the last few scanlines may not be visible in the final video output.)

For interlaced cameras, rolling shutter exists in both fields. An easy way to analyze such video is to deinterlace it first as discussed in Section 3.3.1, then the rolling shutter behaviour can be estimated as a progressive scan camera, with negligible errors [3].

According to the facts above, although a rolling shutter may cause distortion, it does not destroy the basic properties of a continuous exposure setup. Rolling shutter correction is possible and streak based motion capture can still be done. Details about how to adjust every step to work with a rolling shutter camera are shown in corresponding chapters.

3.4 Camera Synchronization

As accurate hardware synchronization is not done for our setup, efforts have to be made during the data processing stage to make inter-camera information meaningful. According to Section 2.5.1, active exposure control is not a good choice for synchronizing streak videos. So we end up with the other way, passive temporal offset measurement, which aims at translating timestamps from different cameras onto a unified common timeline.

Depending on the accuracy needed, such measurement can be divided into two levels: frame level and subframe level. As the names say, the former only cares about the temporal offset among cameras at frame level accuracy, while the latter aims at getting the exact temporal offset between the read-out time of any two specific pixels from two cameras.

Although subframe level accuracy is eventually needed for reconstructing motion, frame level synchronization, which is easier to do, can also be enough for many processing steps. In our system, measurements on the two levels are done separately.

Methodologically, measuring temporal offsets among cameras is essentially the same as measuring when a specific event happens in each camera's own timeline. As the latter version is practically easier to measure, the following discussions focus on this version and concentrate on two subproblems: how to design the event and how to measure when it happens at required accuracy.

3.4.1 Frame Level

A frame level temporal offset measurement can be done by turning on or off a fast-switchable light source, for example a strobe or even a single LED. The time the light source is turned on can be roughly estimated by the timestamp of the first frame which is bright (when synchronizing with a strobe in a dark room) or the first frame with a detectable LED (when synchronizing with a single LED).

The single LED way requires only LED detection, which is robust and can work under any mild lighting condition. Even if there is an error in LED detection, only little human labour is required to fix it manually. Consequently, we decide to use this method in all our algorithms which need frame level synchronization throughout the thesis.

However, it is worth noting that when dealing with rolling shutter cameras, single LED based frame level temporal offset measurement may not be accurate to "one frame". Instead, the error can be as large as two frames (as illustrated in Fig. 3.2). Understanding this is important for designing algorithms based on this measurement.

3.4.2 Subframe Level

Subframe level temporal offset, which provides more information but looks to be much harder to measure, turns out to be measurable in different ways.

An intuitive way is to estimate temporal offsets from stereo streak matching, which can be done during the motion capture process itself. Details

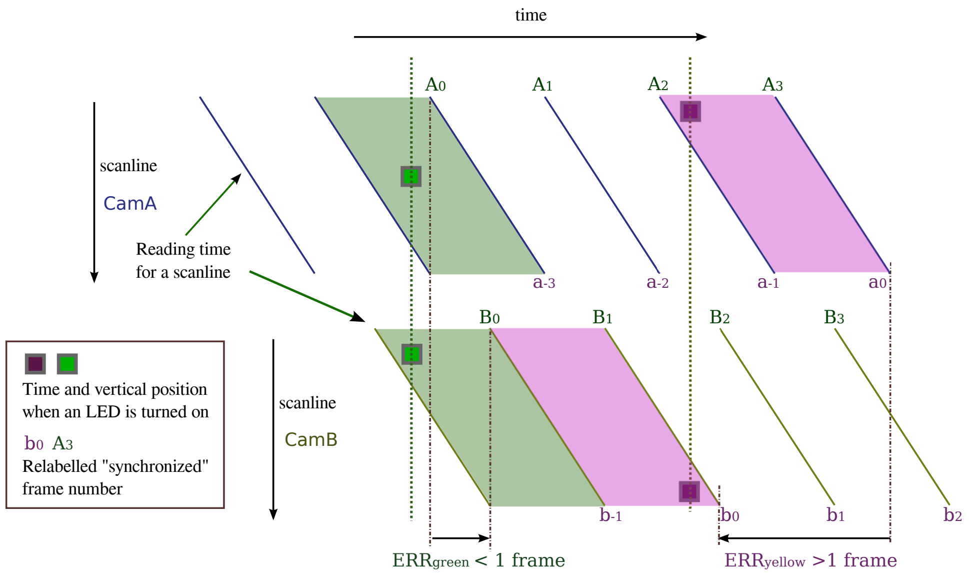


Figure 3.2: Frame level synchronization by an LED and its error with a rolling shutter camera array. Two possible cases are shown in green and purple, respectively. The position of the boxes shows the time and on-image vertical position (corresponding scanline) when the LED is turned on. Synchronization is done by relabelling the first frames with a visible bright LED from each camera (shaded frames in the figure) by frame 0. The green case results in the upper cased relabelling and the synchronization error is under one frame. The purple case, however, results in a relabelling (shown in lower cased letters) with an error larger than 1 frame. Fortunately, from the illustration we can also know that the error will not be larger than 2 frames.

of the method are presented in Section 6.1. As the method has no extra hardware requirement or extra work, it is the preferred way in our system. However, to evaluate this estimation method, a more direct and reliable way to measure subframe level temporal offset is needed.

The more direct method is done in a controlled environment. Specifically, a static scene in a relatively dark room, a fast-switchable LED light source which can dominate the room's illumination, and several cameras in a fixed exposure level (ensuring no saturation occurs when the LED light is on) are needed. The Gamma curve for every camera should also be known so that inverse gamma transform can be done to ensure image intensity is linear to scene irradiance.

With this setup, we start recording videos in the dark and turn on the LED light. The time the light is turned on will be encoded in every video, which once decoded, can be easily translated into temporal offsets for the different cameras.

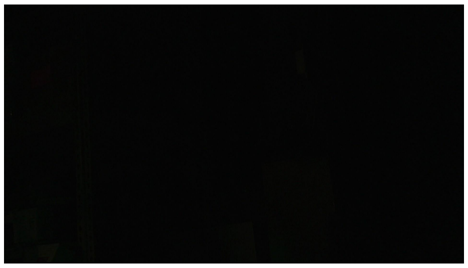
On global shutter cameras, estimating the time is easy. For a video clip from one camera, first get the average intensity for each frame, then we will see some constantly dark frames before some constantly bright frames, with a transition frame in the middle. If we denote the timestamp of the transition image by t_t, the average intensity for dark frames by I_d, the average intensity for the transition frame by I_t, the average intensity for bright frames by I_b, and the exposure interval for each frame by Δ_t, the subframe level timestamp for turning on the light for this camera is:

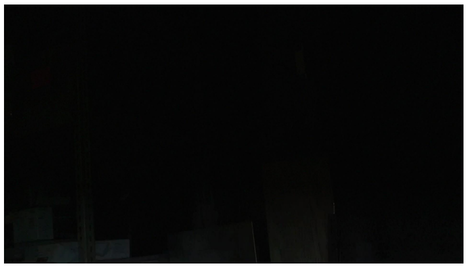
$$T_{\text{light on}} = t_t + \left(1 - \frac{I_t - I_d}{I_b - I_d}\right)\Delta_t \tag{3.2}$$

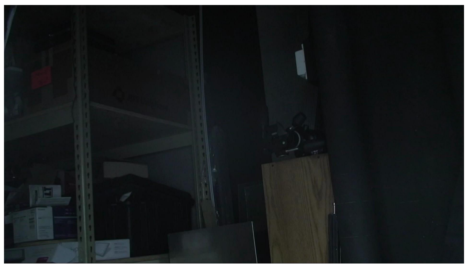
The measurement becomes complex when it comes to rolling shutter cameras. Although there will still be several dark frames and several bright frames, two transition frames instead of one will be found. The temporal offset has to be estimated based on scanline level information rather than frame level. Denote the average intensity of the m-th scanline of frame I_k by $I_k^{(m)}$, the relative intensity of a scanline from a transition image can be defined by:

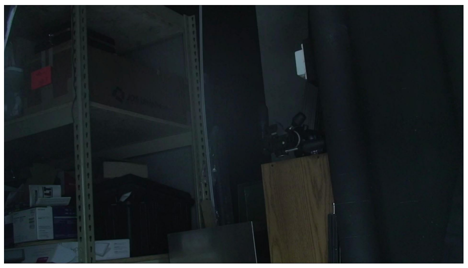
$$I_t^{'(m)} = \frac{I_t^{(m)} - I_d^{(m)}}{I_b^{(m)} - I_d^{(m)}} \tag{3.3}$$

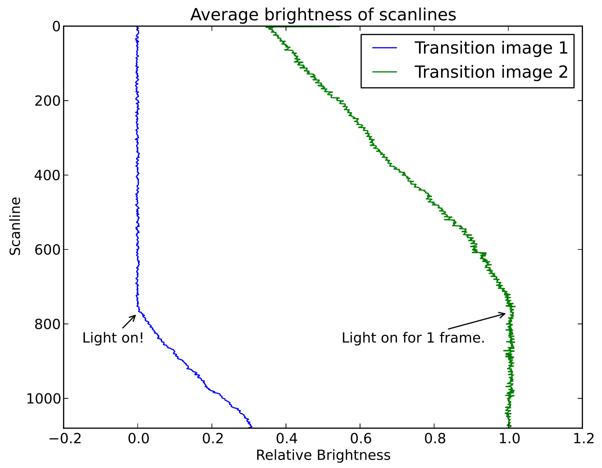
Fig. 3.3 shows the relative intensity of scanlines of the two transition images. From the figure one can easily estimate the first scanline that received light from the light source. The readout time for that scanline is what


(a) A dark frame


(b) Transition frame 1


(c) Transition frame 2


(d) A bright frame



(e) The relative brightness of each scanline

Figure 3.3: With a rolling shutter camera, estimate the camera time when a light is turned on at subframe accuracy. The scene is assumed to be static for at least 3 frames after the light is turned on so that the intensity of a pixel is only related to illumination and exposure.

we are measuring. Assuming it is the k-th scanline of transition frame 1 timestamped t_{t1}, the timestamp for light turned on is:

$$T_{\text{light on}} = t_{t1} + \left(\frac{k}{S}\right) t_e \tag{3.4}$$

where S is the total scanline of current camera mode.

The method is sound and works robustly. However, as the required environment and hardware are not always easily accessible, this method is only used for validation purposes.

3.5 Geometric Calibration of Cameras

In the context of this thesis, geometric calibration refers to finding the projection matrix for each camera in a camera array. Under a pinhole camera model assumption, the calibration includes the measurement of intrinsic and extrinsic parameters of a camera which encode a camera's focal length, position, orientation and other properties.

Compared to traditional stereo reconstruction algorithms based on sharp images, stereo reconstruction on featureless weak-synchronized streaks relies more heavily on epipolar constraints to reduce ambiguity. Accurate calibration of the camera array is desired in our system.

Although lots of research has been done for an accurate and easy way to calibrate cameras (as shown in related work), they either are not able to be applied to camera arrays without accurate synchronization, or require extensive human labour to go through the procedure.

Besides, our camera array setup brings another practical challenge: during calibration, short exposure is generally preferred to avoid motion blur while in the real motion capture we want continuous exposure. However, for most consumer level cameras, adjusting exposure level without physically touching the cameras (which may change a camera's extrinsic parameters) is not possible. As a result, the calibration must be done with the cameras in continuous exposure mode.

One possible solution is to combine the strobe synchronization method ([3], see Section 2.5.1) with CALTag [1], an improved chessboard calibration pattern. As a short synchronized virtual exposure can be achieved, the combination is able to produce sharp rolling-shutter-free images for calibration, which in addition are robust to partial occlusion. However, the reliance on strobes implies the calibration has to be done in a controlled environment and makes the setup much less scalable and portable.

In order to make the calibration process easy to operate while ensuring its accuracy, we adopt the calibration framework from Multi-Camera Self-Calibration ([33], introduced in Section 2.6.3), which calibrates cameras based on synchronized videos of a moving point light source. But instead of relying on a synchronized camera array, we use a blinking LED to synchronize the videos while using the same LED for calibration.

The experiment procedure to do the calibration is extremely simple; all one needs to do is to put all cameras in continuous exposure mode and start recording, then hold a blinking LED in hand and move it around in the experiment space. One thing to notice is that the experiment space should not be too bright so that the detection and localization of the LED can be done robustly.

The principle for synchronization is similar to strobe synchronization [3], but instead of controlling the appearance of the calibration pattern by controlling illumination, our blinking LED method controls the calibration object directly. Similar to [3], our blinking LED based calibration is able to work with continuously exposed rolling shutter cameras without extra synchronization.

The blinking frequency of the LED should be lower than a half of the camera's frame rate (or field rate if the output videos are interlaced), so that any two adjacent frames can only observe at most one LED pulse (as shown in Fig. 3.4). As long as the segmentation of the LED is robust, the on-time of the LED should be as short as possible to avoid motion blur.

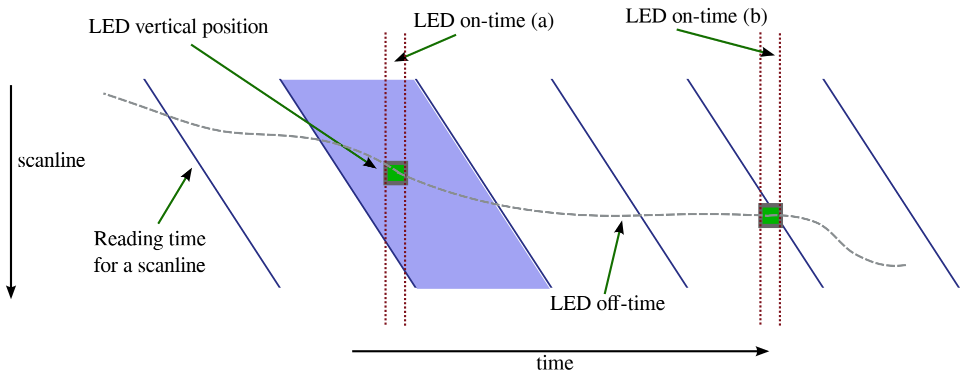


Figure 3.4: Synchronize cameras, remove rolling shutter artifacts and reduce motion blur by a blinking LED for accurate geometric calibration. The LED may appear in a single frame (LED on-time a) or two consecutive frames (LED on-time b), but will never appear in three consecutive frames.

A compact and portable blinking LED can be built with 555 timer IC chip (details and the circuit diagram can be found in Appendix A). However, as the output frequency for a 555 chip is not perfectly stable, to avoid temporal ambiguity in later processing, a blinking frequency lower than one fourth of the frame rate is preferred.

For software implementation, the recorded videos from different cameras are first synchronized at frame level by the first LED blink, and the timestamp of each frame is relabelled accordingly. Then, frames with one detectable LED are extracted with their timestamps. Note that if an LED is found in two adjacent frames, they should be at the same position (see Fig. 3.4) and either frame can be used. Next, the extracted frames are grouped by their timestamps so that the difference of frame timestamps in a group should not be larger than the frame level synchronization error (1 frame for global shutter and 2 frames for rolling shutter as shown in Fig. 3.2).

Now each group of frames is essentially synchronized shots of the LED. The rest of the work, such as localization of an LED in an image, outlier removal, projection matrix estimation and bundle adjustment, will all be exactly the same as in Multi-Camera Self-Calibration.

An additional note is that to make frame level synchronization work, during the first blink the LED should be positioned visible to all the cameras. After that, the LED can be moved freely and can be out of the views of some of the cameras because the grouping step is based on timestamps instead of the number of blinks observed. When calibration is finished, blinking LEDs are not used any more.

Overall, the proposed blinking LED based calibration method is an easy, cheap and portable method that can work with unsynchronized rolling shutter cameras and it inherits all the advantages of Multi-Camera Self-Calibration.

Chapter 4

Streak Segmentation and Track Extraction

The main goal of this chapter is to extract 2-D projected tracks from streaks in a video clip.

In our framework, this step is done independently for each camera view, so all processing and algorithms in this chapter are limited to information from a single camera. Errors and ambiguities are allowed in this step, which will hopefully be identified and resolved in later steps where multi-view information is available.

4.1 Overview

4.1.1 Assumptions

Our track extraction algorithm is based on the following basic assumptions on the light sources and motion we are capturing:

- **Mechanical Property.** All light sources are assumed to be real-world objects which obey Newton's laws of motion. In other words, only the acceleration of a light source can be changed abruptly, while the speed and position of a light source should always be continuous.
- **Optical Property.** Optically, light sources are assumed to be bright and small, and should reflect or emit light in a constant manner. When captured by cameras, such static light sources should appear circularly symmetric.
- **Simple Trajectory.** The projected trajectory of a light source is assumed to be "simple" (not self-intersecting) within the time span of any single frame.

These assumptions are made mainly to make the problem better defined and easier to solve. Practically, we try to design algorithms which are robust to slight violations of these assumptions. Especially for this chapter, we try hard to "fail gracefully" so that some failures can be corrected automatically in later steps when information from more camera views is available.

4.1.2 Definitions

As pointed out in the introduction, the word "streak" refers to the bright thick curves on images left by a moving point light source.

A 2-D track is a projected motion trajectory of one marker. Formally, a **track** $\boldsymbol{s}$ can be described by a set of pixels labelled with timestamps:

$$\boldsymbol{s} := \{(x_k, y_k, t_k) | k \in \{0, 1, 2, \ldots\}\}, \tag{4.1}$$

in which x_k, y_k are image coordinates and t_k is the approximated timestamp on the timeline of the video where the track is extracted from. Note that the timestamps should already be aligned at frame level.

Initially, tracks are extracted from only one frame. But because of continuous exposure, tracks from consecutive frames can be connected into a **long track** if they belong to a same object. Formally, a long track is essentially the same as a track.

All the long tracks from a **video clip** forms a set:

$$\boldsymbol{S}^{(i)} := \{\boldsymbol{s}_1^{(i)}, \boldsymbol{s}_2^{(i)}, \ldots\}, \tag{4.2}$$

where i refers to the camera recording the video clip. For simplicity, we assume for each experiment, only one video clip is recorded for each camera.

Given the formal definitions, the goal of this chapter is to find the long track set $\boldsymbol{S}^{(i)}$ for a given video clip.

4.1.3 Algorithm Overview

A diagram of the whole track extraction procedure is shown in Fig. 4.1. First, streaks are segmented from each frame. Then streaks from consecutive frames are analyzed to get per image tracks with timestamps. Finally, tracks from different frames are connected to long tracks. Arrows in the diagram show what data is needed for each step.

The details of each step are discussed in the following sections.

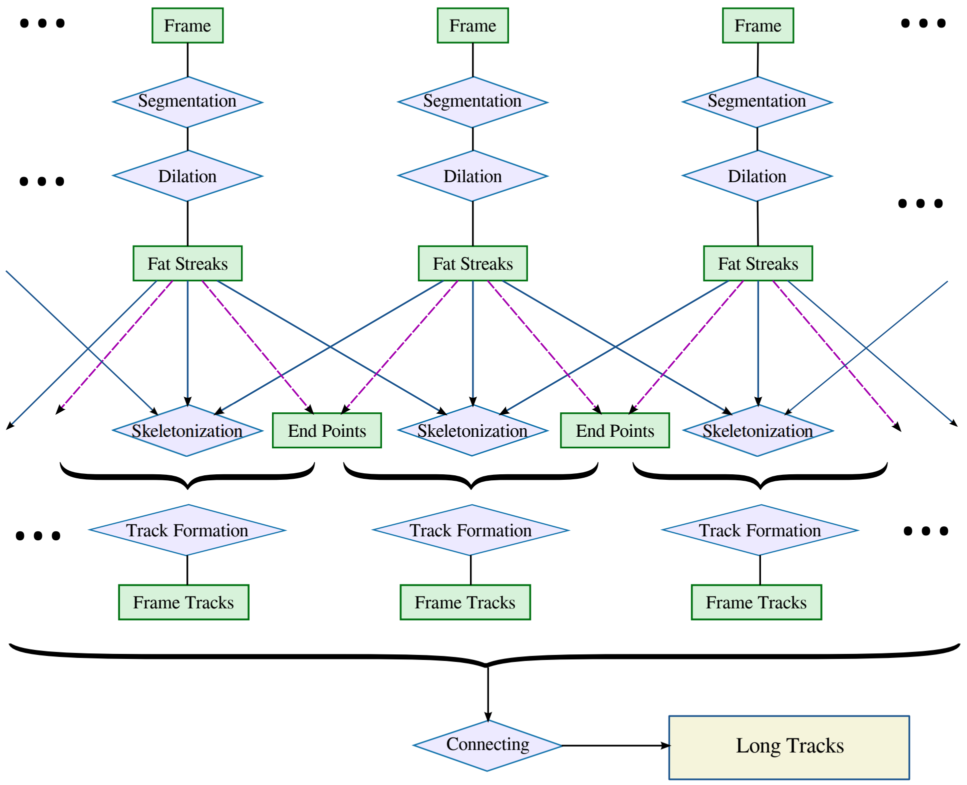


Figure 4.1: An overview of the whole procedure to extract long tracks from a video clip. See text for details.

4.2 Frame Tracks

4.2.1 Streak Segmentation

First of all, streaks have to be segmented from the frames. This can be challenging because during the long exposure, fast moving markers will become dimmer and might even be blended into the background.

To segment streaks robustly, one can either develop complex segmentation methods to deal with dimmed marker streaks, or increase the brightness of the light sources so that they can always be easily segmented even under fast motion. We believe the first way should be possible if careful analysis of the color and shape of streaks is done, but for simplicity, in our work we choose the second way, which implies a simple segmentation method but a careful design of the light sources (markers).

With bright light sources and a proper camera setting, the light sources are assumed to saturate the sensors. With this assumption, a simple segmentation algorithm can be done by the following steps:

1. Background reduction.
2. Simple thresholding with a very high threshold.
3. Do morphological opening to get rid of outliers.

Note that although a high recall rate is desired to avoid missing streaks, the precision does not have to be very high because the steps to form tracks and stereo matching will automatically get rid of most false positives in segmentation.

4.2.2 Endpoint Detection

Endpoints of streaks in each frame are important because they are the only points on a streak that can be reliably timestamped. A robust way to extract such endpoints is desired.

As the shape of a streak can vary largely and segmentation results may suffer from noise, it is very hard to find a general method to robustly extract endpoints from a single streak.

But the problem becomes much easier when streaks from two consecutive frames are available. Due to continuous exposure, streaks from two consecutive frames should always be connected, and the connection points are naturally the endpoints of the streaks connected to it. Further, we even

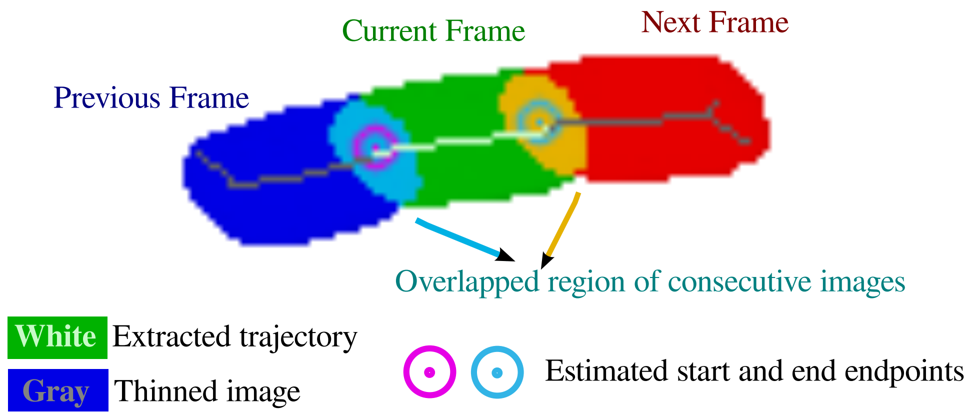

Current Frame
Next Frame
Previous Frame
Overlapped region of consecutive images
White Extracted trajectory
Gray Thinned image
Estimated start and end endpoints

Figure 4.2: An illustration of the track extraction procedure. The segmented streak of the previous, current and next frame are shown in blue, green and red blobs, respectively. The three blobs are overlaid and thinned, producing the dark grey track (note the branches at the ends of the track). To extract the track for the current frame, the endpoints of the track for the current frame are estimated by overlapping blobs of every two consecutive frames. The final extracted track is the part between the two endpoints (shown in white).

know that the position of a connection point actually corresponds to the position of the light source at the read-out time of the earlier frame.

Practically, because the light source has a visible size, the exposure of the pixels around the connection point are spread to both frames. The outcome is that streaks of a light source from two consecutive frames overlap around the connection point, and the overlapped regions are dimmer (unless the sensor is saturated) than the rest parts of the streaks.

However, as the overlapping regions can sometimes be too dim to be segmented out, some endpoints might be missed by simply overlapping the segmented blobs. Lowering the intensity threshold for segmentation may solve the problem in some occasions, but also will make the segmentation result noisy. On the other hand, we find that morphological dilating the segmented result solves the problem robustly, without changing the shape of streaks.

Our final endpoint detection algorithm:

- Do morphological dilation on all the segmented frames.
- Find all the overlapping blobs from every two consecutive segmented frames. Each blob is then a possible streak endpoint.
- The position of the endpoint is estimated by the centroid of the blob.
- The timestamp of a possible endpoint is the read-out time of the centroid pixel in the first of the two frames.

An illustration of this procedure is shown in Fig. 4.2, the two endpoints are shown in circles.

4.2.3 Streak Skeletonization

Due to glare effects, streaks are typically thick. To get the actual projected trajectories of the light sources, a thin skeleton has to be extracted from every fat streak.

Based on the assumption that glares and shapes of light sources are symmetric in all directions, the medial axis of a streak should be a good approximation for the projected trajectory.

A common and efficient way to extract medial axis from a given shape is to detect ridges in the distance map of boundary points [29]. However, as the topological structure is not maintained by this method, the output skeleton might be discontinuous. So we do not adopt this method.

Instead, a skeletonization algorithm based on iterative thinning is used. Each thinning operation is implemented by hit-and-miss transformation [13]. To reduce noise, morphological opening operation is applied before thinning, and morphological pruning operation [13] is done afterwards.

Due to the nature of the thinning algorithm, the endpoints of a streak are very likely to be split into two branches after skeletonization. If any of the branches happens to be included in the final track, an error will occur in motion reconstruction. However, as the appearance of the branches can vary largely depending on the shape of the end of streaks, direction of motion and the angle of view (see examples of branches in Fig. 4.2), it is very hard to find a robust way to remove the branches as a post-processing step.

To work-around the problem, as shown in Fig. 4.1 and Fig. 4.2, for each frame, the two frames before and after it are overlaid onto it before starting thinning operation. As a result, the endpoints of streaks in a frame are no longer the endpoints in the image to be thinned, so the branching problem is solved automatically.

The overlaying operation also reduces the chance of thinning the streak in a wrong direction for a slowly moving object because overlaid streak tends to be longer.

4.2.4 Track Formation

Given all the potential endpoints and skeletons, the next step is to combine them to form tracks.

Apparently, endpoints of a track should lie on the track. However, as the endpoints and skeletons above are extracted separately via different methods, it is possible that endpoints may not lie on any skeleton. We assume the actual position of an endpoint is a pixel fulfilling the following three requirements: (1) the pixel should be on a skeleton; (2) the pixel must belong to the same blob in the segmented image; (3) among the pixels fulfilling the two previous requirements, it should be the nearest one to the endpoint.

The next step is to find out the correspondence among start endpoints, end endpoints and skeletons so that tracks can be extracted from each frame. This is not trivial because: (1) more than two endpoints might be on one skeleton due to various reasons, including streak intersection; (2) skeletons may have branches because of noise.

To form tracks robustly, we run a Breadth-First-Search (BFS) from each start endpoint along the skeleton it is attached to. Most likely, one and only one end endpoint will be found on the same skeleton and the shortest path between the two endpoints along the skeleton will be obtained. In this case, the two endpoints and the shortest path forms a track. An example of this case is shown in Fig. 4.2.

If no end endpoint or more than one endpoint are found along one single skeleton, it means that something bad has happened (see Fig. 4.3). At this time one may do some complex analysis on all possible tracks and filter away the ones with sharp turns (which imply a huge acceleration). But as such complex analysis may not be robust as well, some simple workarounds may also be considered: either to discard the whole skeleton, or to extract all possible tracks from the streak. The former is recommended for a multi-camera setup because it is possible that nice tracks will be acquired from other views if the two trajectories are not actually intersecting in the 3-D space, while the latter one is a good choice if not enough cameras are available and there is still a good chance that stereo matching will automatically choose those streaks that actually corresponds the trajectory of a real light source. In our implementation, we choose to discard all problematic streaks.

A newly formed track only has timestamps on its endpoints. To provide more information for stereo matching, we estimate a timestamp for every pixel along a track. The estimation can be done by any interpolation methods. As the estimation does not directly apply to the final output, even a linear interpolation should work well enough.

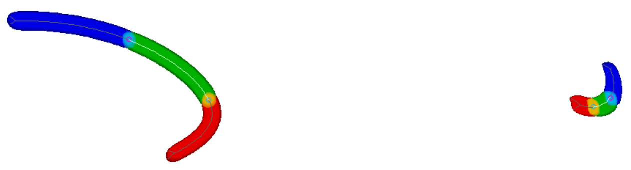
4.3 Long Tracks

Frame tracks can already be used for stereo matching, but they suffer from several deficiencies:

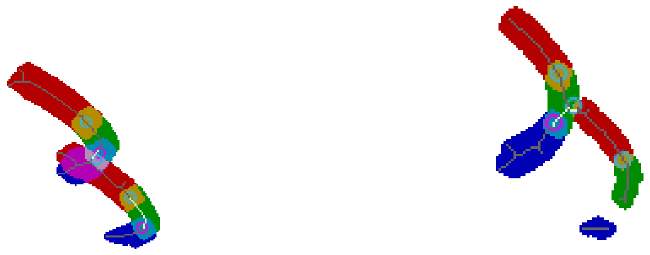
- When processed independently, the relationship between frame tracks is hard to describe and utilize.
- As cameras are only weak-synchronized, a frame track from one camera cannot be directly matched to tracks from another camera, because they are not exposed in the same time interval.
- Frame tracks are typically short. In the stereo matching step, such short track might be matched to a wrong track by accident.

Our solution is to connect the tracks of one light source from consecutive frames to a longer track. This can be easily done because the video has continuous exposure.

Figure 4.3: Examples for successful and failed track extraction.

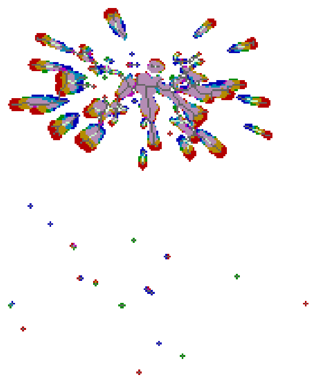


(a) Success cases.



(b) Failure Case: the two streaks intersect each other.

(c) Failure Case: intersecting streaks and a discontinued streak.



(d) A complex case from a video of fireworks[a]. The isolated points are noise (most of which are detected when frames are overlaid). The top half of the figure shows many examples of track intersection. Different failure handling methods will lead to different track extraction outputs.

[a]To test the spatial scalability of the system and its performance in a multi-marker scenario, we tried to reconstruct 3-D motion of fireworks. Unfortunately, due to the huge crowd in the fireworks event, we were not able to setup all the cameras as planned and the experiment failed eventually.

To connect frame tracks to long tracks, we process all the frames sequentially by time. For each frame, find all the tracks that can be connected a track from the last frame and connect them. However, an important factor that needs to be taken into consideration is the potential errors.

There are two types of errors: one type is that two tracks that should be connected are not connected; the other type is that tracks of different light sources are connected by mistake. The outcome of the two types of errors are also quite different. The former does not harm too much because it just means some information will not be well utilized. The latter, however, is critical because all later algorithms based on the assumption that a long track corresponds to one single light source will fail.

As a result, we put strict constraints when trying to connect tracks from consecutive frames:

- If the distance between the end of the first track and the start of the second track is larger than a threshold, then the two tracks can't be connected.
- If more than one track from one frame can be connected to a track in a neighbour frame, do not connect any of them.

With these constraints, "false positive" connections are effectively limited so that in most cases, we can assume each long track corresponds to at most one point light source.[1]

In the following chapters, "long tracks" are referred as "tracks" for simplicity.

[1]A possible violation of this assumption is that the projection of two light sources might overlap in a specific view. We discuss this in Section 8.1.2.

Chapter 5

Stereo Track Matching

A basic problem in 3-D motion reconstruction is 3-D motion trajectory reconstruction. Like most multi-view reconstruction tasks, we do it based on stereo matching and triangulation. Specifically, instead of matching each point on tracks independently, we do stereo matching on the level of tracks (the pixel level matching is only done during track matching). We choose this framework because of the following reasons:

- Considering the assumption that each extracted track corresponds to only one point light source, points in a track are closely related and should be processed as a whole, so that the relation among points in a track can be utilized.
- Efficiency. If two tracks can be quickly found to be impossible to be matched, one does not have to match the whole tracks.
- Robustness. If two tracks are found impossible to match as a whole, all candidate point matchings between the two tracks will be known as invalid and can be ruled out.

In this chapter, we present our method to match two tracks from two different camera views.

5.1 Ambiguity

The core challenge for matching tracks is ambiguity, which exists by nature and is magnified by quantization and measurement errors.

Generally, stereo matching algorithms can make use of the following constraints:

- **Epipolar Constraints.** Under a pin-hole camera model, if the projection of a 3-D point in one view is given, the projection of the same point in another camera view must lie on a known line, namely the epipolar line or epiline.

- **Temporal Constraints.** Correctly matched pixels should correspond to a same 3-D position. In a dynamic scene, this not only requires the two pixels correspond to a same object, but also the same object at a specific time.

- **Local Appearance Cue.** The appearances of the neighbourhoods of a point's projections on different camera views should look similar.

The first two constraints are applicable to streak image matching as well. However, the local appearance cue can hardly help matching featureless streak images, which are quite different from the featureful images matching in standard stereo image matching. As a result, ambiguity, which refers to the fact that more than one point in one view can be matched to a point in another view according to the constraints above, is more likely to occur.

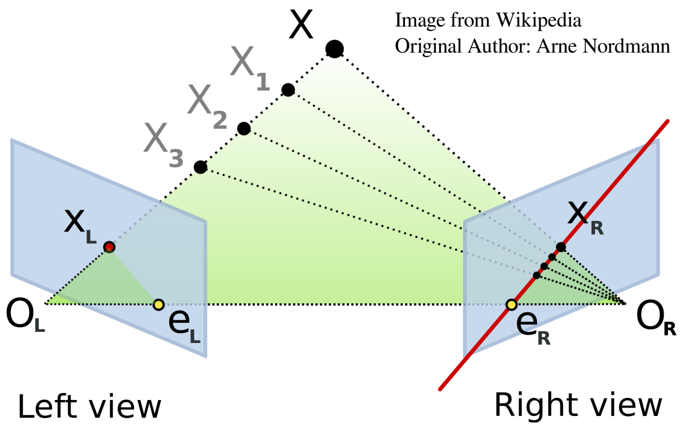


Figure 5.1: Epipolar geometry and binocular stereo matching ambiguity. If a point light source's 3-D trajectory lies within the O_LO_RX plane, its projected 2-D tracks on the two views will be simple segments on the epilines ($X_L - e_L$ and $X_R - e_R$). Reconstructing the actual 3-D trajectory only from these two views is theoretically not possible.

For a binocular stereo system, an illustration of the ambiguity is shown in Fig. 5.1. If a light source moves within the epipolar plane (the plane defined by O_L, O_R and X in the figure), its tracks will simply be on the epipolar lines of the images. Without an accurate timestamp on every track point, it is theoretically impossible to correctly match the points on the two tracks.

Although in reality it is almost impossible for a random motion in 3-D space to be constrained exactly in the epipolar plane, motions near the epipolar plane will also cause ambiguity in real experiments due to calibration errors and sub-sampling during digitalization. A real example is shown in Fig. 5.2, where the 2-D tracks of a point on a spinning fan are being matched. From the figure, we can see that lots of pixels satisfying the temporal constraint are lying in the 2-pixel neighbourhood of the epiline, which is a reasonable upper limit for calibration error.

The ambiguity problem can potentially be solved in two ways: regularization or multi-view stereo.

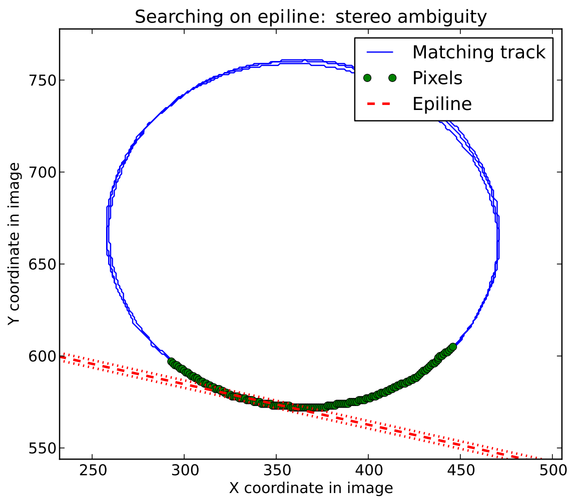
Regularization guesses the best among candidate matchings by using prior knowledge, such as the continuity and smoothness of the reconstructed 3-D trajectory. The method requires no extra information to be measured and usually gives reasonably good guesses, but is not guaranteed to be right every time.

Multi-view stereo solves ambiguity by introducing multi-view consistency check, which brings stronger geometry constraints. When it is able to solve the ambiguity problem, it can solve it accurately. However, the method has many problems:

- As no identification on light sources can be done, multi-view consistency might be falsely satisfied by other light sources by coincidence.
- A light source might be occluded in some of the views. In the worst case, multi-view consistency check will discard tracks visible only to two cameras, which might be processable by regularization.
- It is difficult to put a threshold on the number of consistent camera views needed to ensure a correct matching.
- More cameras have to be deployed during the capture process.
- The computation cost of multi-view consistency check can be huge.

Considering the pros and cons, we make a combination between the two. The problem will mainly be dealt by regularization, but it can also take multi-view information into account when such information is available and appears to be necessary.

To present our method, we will first define the stereo track matching problem formally, and introduce our matching algorithm based on that.



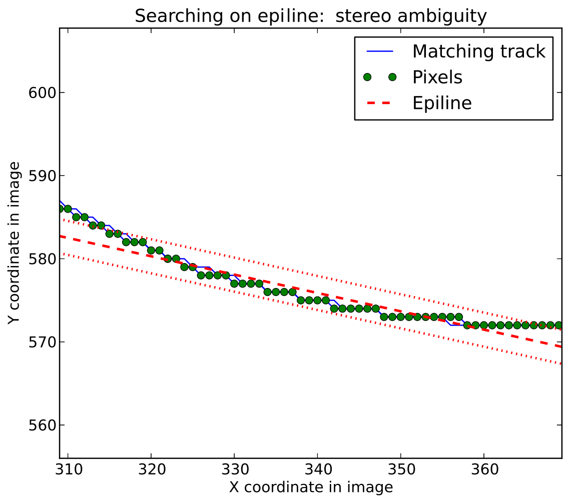


Figure 5.2: A real example showing the ambiguity in stereo track matching. The motion behind the example is an LED mounted on a blade of a spinning fan. The task being done is to find a point on a track (shown in blue) to match a point from another camera view. Pixels satisfying the temporal constraints are shown as green circles. The red dashed line is the epiline, and the two dotted red lines show the two pixel neighbourhood of the epiline. One can see that lots of candidate points satisfy both the epipolar and temporal constraints, causing a matching ambiguity. The bottom figure is a zoomed version of the top one. Raw frames from this experiment can be found in Fig. 7.5.

5.2 Formal Definition

The pairwise track matching problem can be formulated as: given two tracks extracted from two different cameras, $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, how can they be matched by stereo and if they match, how exactly do they match at the pixel level. To address this problem, a quantitative measure of their matching cost, $c(\boldsymbol{s}_1, \boldsymbol{s}_2)$, is also required. The two cameras are assumed to be fully calibrated, and the projection matrices are denoted by P_1 and P_2, respectively.

First of all, if two tracks can be matched, they must have an exposure interval in common. Actually, only those parts of tracks from the common exposure interval may affect the matching cost. So first the two tracks are sliced to keep only the commonly exposed part (in practice, a small extra part needs to be preserved due to the weak frame level synchronization):

$$\boldsymbol{s}'_1 := \{(x,y,t) | (x,y,t) \in \boldsymbol{s}_1 \wedge (\min t_{\boldsymbol{s}_2} - e_{\text{sync}} \leq t \leq \max t_{\boldsymbol{s}_2} + e_{\text{sync}})\}$$
$$\boldsymbol{s}'_2 := \{(x,y,t) | (x,y,t) \in \boldsymbol{s}_2 \wedge (\min t_{\boldsymbol{s}_1} - e_{\text{sync}} \leq t \leq \max t_{\boldsymbol{s}_1} + e_{\text{sync}})\}$$

where $\min t_{\boldsymbol{s}_i}$ refers to the minimum timestamp for pixels on track $\boldsymbol{s}_i$ and e_{sync} is the maximum possible error on frame level synchronization.

Now, if the two tracks do not have any part that was exposed in a common interval, we can already conclude that they do not match. Otherwise we can concentrate on the sliced part:

$$c(\boldsymbol{s}_1, \boldsymbol{s}_2) = \begin{cases} \infty & \boldsymbol{s}'_1 = \emptyset \vee \boldsymbol{s}'_2 = \emptyset \\ c(\boldsymbol{s}'_1, \boldsymbol{s}'_2) & \text{otherwise} \end{cases} \tag{5.1}$$

To define $c(\boldsymbol{s}'_1, \boldsymbol{s}'_2)$, we model the matching between two tracks as a pixel labelling scheme: each pixel on one track (the reference track) is labelled by at most one matched pixel on the other track. Formally, a labelling scheme can be described as a function from the reference track to the other track:

$$\mathcal{L}_{a,b} : \boldsymbol{s}'_a \to \boldsymbol{s}'_b \cup \{\emptyset\}, \quad a, b \in \{1, 2\} \tag{5.2}$$

If we can define a cost function (or energy function) for each pixel labelling scheme $E(\mathcal{L})$, the best labelling scheme can be described as:

$$\mathcal{L}_{\text{best};a,b} = \operatorname*{arg\,min}_{\mathcal{L}_{a,b}} E\left(\mathcal{L}_{a,b}\right) \tag{5.3}$$

Note that pixel labelling is not symmetrically defined, to get symmetric costs for track matching, a simple but reasonable solution is to average the cost both ways:

$$c(\boldsymbol{s}'_1, \boldsymbol{s}'_2) = c(\boldsymbol{s}'_2, \boldsymbol{s}'_1) := \frac{E(\mathcal{L}_{\text{best};1,2}) + E(\mathcal{L}_{\text{best};2,1})}{2}$$

As the order of points on a track can be important in matching, we write each track as a sequence of pixels, ordered by their estimated timestamps:

$$\boldsymbol{s}'_a = (x_{a,1}, y_{a,1}, t_{a,1}), (x_{a,2}, y_{a,2}, t_{a,2}), \cdots \qquad i < j \Leftrightarrow t_{a,i} < t_{a,j} \tag{5.4}$$

5.3 Energy Function

To actually evaluate and optimize the cost of a potential matching between two tracks, an energy function needs to be defined. Intuitively, the following factors should be reflected in the energy function:

- **Epipolar Constraints (e).** The basic geometrical constraint for stereo matching. Due to errors in calibration and track extraction, the constraints may not be perfectly satisfied, so some relaxation is needed when applying this constraint.

- **Time Constraints from Frame-level Synchronization (t).** As the timestamps are already synchronized at frame level, the difference between the timestamps of two matched pixels should always be smaller than the maximum possible error of frame-level synchronization.

- **Smoothness (s).** Ideally, according to Newton's Laws of Motion, the 3-D trajectory reconstructed from matched tracks should be continuously differentiable. However, due to discretization in digital sampling, trajectories can only approximately be smooth.

- **Temporal Offset Coherency (toc).** As the offset between the timelines of two video clips is always constant, the sub-frame level differences between the timestamps of matched pixels shall not vary abruptly.

- **Multi-View Consistency (m).** When more than two camera views are available, multi-view geometry constraints can be used to validate a matching.

Correspondingly, the overall cost function can be written as a sum of five cost functions:

$$E(\mathcal{L}_{a,b}) := E_e\left(\mathcal{L}_{a,b}\right) + \lambda_t E_t\left(\mathcal{L}_{a,b}\right) + \lambda_s E_s\left(\mathcal{L}_{a,b}\right) + \lambda_{\text{toc}} E_{\text{toc}}\left(\mathcal{L}_{a,b}\right) + \lambda_m E_m\left(\mathcal{L}_{a,b}\right) \tag{5.5}$$

where $\lambda_t, \lambda_s, \lambda_{\text{toc}}, \lambda_m$ are constant coefficients which can be manually decided to balance the weight of different factors.

Each energy term can be expanded to a sum of pixel-level energy terms. To simplify representation, the position and timestamp of a pixel matched to the k-th pixel of track $\boldsymbol{s}'_a$ under matching scheme $\mathcal{L}$ will be referred to as:

$$(x_{\mathcal{L},k}, y_{\mathcal{L},k}, t_{\mathcal{L},k}) := \mathcal{L}(\boldsymbol{s}'_{a,k}) \tag{5.6}$$

A special case is that some pixels may have no match in the other track. In such case we will assign a constant energy to every term associated with the null match.

5.3.1 Epipolar Constraint

The quantitative measure of how well the matched pixels fit the epipolar constraint is defined by the on-image-distance of a pixel to the epiline of the other pixel. Formally, given the projection matrices P_a, P_b of the two cameras, the distance from a candidate matching pixel $(x_{\mathcal{L},k}, y_{\mathcal{L},k})$ to the epiline of the reference pixel $(x_{a,k}, y_{a,k})$ can be computed and denoted by $d_e(P_a, P_b, (x_{a,k}, y_{a,k}), (x_{\mathcal{L},k}, y_{\mathcal{L},k}))$.

One way to define the epipolar constraint energy of a pixel matching is as follows:

$$e_e(P_a, P_b, (x_{a,k}, y_{a,k}), (x_{\mathcal{L},k}, y_{\mathcal{L},k})) := \begin{cases} d_e & d_e < \epsilon_d \\ \infty & \text{otherwise} \end{cases} \tag{5.7}$$

where ϵ_d is a threshold which may depend on the accuracy of camera calibration.

With this definition, the overall epipolar constraint term can be written as the sum of energies of every pixel matching:

$$E_e(\mathcal{L}_{a,b}) = \sum_k e_e\left(P_a, P_b, (x_{a,k}, y_{a,k}), (x_{\mathcal{L},k}, y_{\mathcal{L},k})\right) \tag{5.8}$$

5.3.2 Time Constraint

After frame-level synchronization, timestamps of matched pixels should never have a large difference:

$$e_t(t_{a,k}, t_{\mathcal{L},k}) = \begin{cases} \infty & |t_{a,k} - t_{\mathcal{L},k}| > \epsilon_t \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

where ϵ_t is a threshold, which can be the largest possible error of frame-level synchronization.

Similarly, the overall energy defined by time constraint is:

$$E_t(\mathcal{L}_{a,b}) = \sum_k e_t\left(t_{a,k}, t_{\mathcal{L},k}\right)$$

5.3.3 Smoothness

As smoothness is defined on the reconstructed 3-D trajectory, triangulation is done for each candidate matching. Denote the triangulation result for $(x_{\mathcal{L},k}, y_{\mathcal{L},k})$ and $(x_{a,k}, y_{a,k})$ by $\boldsymbol{x}^*_{\mathcal{L},k}$, the smoothness constraint can be described as:

$$E_s(\mathcal{L}_{a,b}) = \sum_k \|\boldsymbol{x}^*_{\mathcal{L},k} - \boldsymbol{x}^*_{\mathcal{L},k+1}\|_2^2$$

5.3.4 Temporal Offset Coherency

The difference of timestamps of correctly matched pixels should remain constant. However, because most timestamps on a track are only estimated by interpolation, we only require the difference of timestamps of matched pixels change smoothly:

$$E_{\text{toc}}(\mathcal{L}_{a,b}) = \sum_k |(t_{\mathcal{L},k} - t_{a,k}) - (t_{\mathcal{L},k+1} - t_{a,k+1})|^2$$

5.3.5 Multi-View Consistency

When multiple camera views are available, one can utilize the consistency among different views to validate matches. One way to do this is to project the reconstructed point into other camera views and check if there is a track passing through or near the projected point.

To do this quantitatively, for each other camera, the smallest distance between the projected point and any track point satisfying the temporal and geometrical constraint in that camera view needs to be computed. Then, considering the fact that the light source might be occluded in some of the camera views, only the p smallest "smallest distances" are averaged to represent how well multi-view consistency is reached. (The choice of p depends on the accuracy needed and the number of cameras available. $p = 2$ is probably a good choice because this implies that at least 4 cameras have to be consistent on this reconstruction.)

Note that this procedure can be computationally extremely intensive. Our experiments also show that in most cases this does not make a big difference on the final result, unless there is a severe ambiguity going on which is hard to resolve by later steps.

Now the energy function is well defined. As long as the related computation and optimization can be done, a "best" track matching scheme and a corresponding reconstructed 3-D track will be obtained.

5.4 Computation

5.4.1 Distance to Epiline

To compute the distance $d_e(P_a, P_b, (x_{a,k}, y_{a,k}), (x_{\mathcal{L},k}, y_{\mathcal{L},k}))$ from a candidate matching point to the epiline of the reference point, the first step is to find the epiline.

A standard way to find an epiline is to preprocess tracks by stereo rectification, then the epiline of a given point is just the same horizontal line in the other image, which can be computed efficiently. However, rectification is not good for our track matching task because:

- Largely varying view angles make it impossible to reproject all the camera views onto one single plane.
- Rectification warps images. Distances to epilines of different points become uncomparable and not directly related to calibration error.

Although the procedure to find epilines without rectification is more complex and slower, it has been well developed and is included in libraries like OpenCV. Further, efficiency for computing a single epiline is not a big problem because for static cameras precomputation can be done if needed, which has essentially the same computation cost as rectification.

Once an epiline l described by $ax + by + c = 0$ is found, the distance from pixel (x_0, y_0) to the epiline can be obtained by $|ax_0 + by_0 + c|$ when $a^2 + b^2 = 1$.

5.4.2 Triangulation

Triangulation is required to get reconstructed points $\boldsymbol{x}^*_{\mathcal{L},k}$ for the smoothness energy term. As described in [14], it can be done by solving linear equations.

Given the position of matched pixels in the two views $\boldsymbol{x}_1, \boldsymbol{x}_2$ and the projection matrices P_1, P_2 (all in homogeneous coordinates), the position of the 3-D point $\boldsymbol{X}$ should satisfy $\boldsymbol{x}_1 = P_1\boldsymbol{X}, \boldsymbol{x}_2 = P_2\boldsymbol{X}$. To eliminate the homogeneous scale factor, write the equations in cross product: $\boldsymbol{x}_1 \times (P_1\boldsymbol{X}) = 0, \boldsymbol{x}_2 \times (P_2\boldsymbol{X}) = 0$.

The two cross product equations can be rewritten into matrix form:

$$A\boldsymbol{X} := \begin{bmatrix} x_1\boldsymbol{p}_1^{3T} - \boldsymbol{p}_1^{1T} \\ y_1\boldsymbol{p}_1^{3T} - \boldsymbol{p}_1^{2T} \\ x_2\boldsymbol{p}_2^{3T} - \boldsymbol{p}_2^{1T} \\ y_2\boldsymbol{p}_2^{3T} - \boldsymbol{p}_2^{2T} \end{bmatrix} \boldsymbol{X} = 0 \tag{5.10}$$

where $\boldsymbol{p}_1^{3T}$ means the third row of matrix P_1. The equation is an over determined homogeneous equation. According to [14], its least-squares solution, which minimizes $||A\boldsymbol{X}||$ subject to $||\boldsymbol{X}|| = 1$, is the last column of V, where $A = UDV^T$ is the singular value decomposition of A.

The above method implies an SVD decomposition for every possible matching, which can be slow. If performance is a major concern, one can first project a pixel to the epiline and then do triangulation with the projected point. Then the equation will not be over-determined and can be solved efficiently.

5.5 Optimization

On carefully investigating each term of the overall energy function, we find that it can be modeled as a 1-D sequence labelling problem under a Hidden Markov Model (HMM) [30]:

- Pixels on the reference track form an **observed sequence**.
- Each pixel on the other track can be viewed as a **state**.
- The **output probability/energy** (probability of an observation given a underlying state) can be defined by energy function e_e and e_t.[2]
- The **transition probability/energy** can be defined by energy function E_s and E_{toc}.

Then, the best labelling and its cost can be efficiently computed using a standard HMM solver with Viterbi algorithm, a dynamic programming based algorithm with a theoretical complexity of $O(n_{\text{observation}} \cdot n_{\text{state}}^2)$. In

[2]A non-negative energy is a common alternative way to describe probability. The relationship between the two is $p = \exp(-e)$. Maximizing probability is equivalent to minimizing energy.

the case of finding the best matching $\mathcal{L}_{\text{best};a,b}$ between tracks $\boldsymbol{s}'_a$ and $\boldsymbol{s}'_b$, the theoretical complexity is $O(|\boldsymbol{s}'_a| \cdot |\boldsymbol{s}'_b|^2)$, in which the output energy is calculated $O(|\boldsymbol{s}'_a| \cdot |\boldsymbol{s}'_b|)$ times and the transition energy is computed $O(|\boldsymbol{s}'_a| \cdot |\boldsymbol{s}'_b|^2)$ times. Although the complexity is polynomial, the order is still relatively high, making it practically running slow.

To improve the efficiency of the algorithm, we note that many state-observation pairs (stereo pixel pairs) have infinite energy because the corresponding pixels are impossible to be matched due to epipolar or time constraints. Pruning such states will effectively reduce the computations for the transition energy term. The new complexity can be written as $O(|\boldsymbol{s}'_a| \cdot |\boldsymbol{s}'_b| + |\boldsymbol{s}'_a| \cdot \alpha^2)$, where α is the number of possible matches for a pixel on average, which is typically a small number.

Still, the $O(|\boldsymbol{s}'_a| \cdot |\boldsymbol{s}'_b|)$ term can be large when both tracks are very long. Fortunately, this number can also be effectively reduced by utilizing the frame-level time constraint. Every time we need to find candidate matches for a pixel, we only need to search among the pixels on the other track with similar timestamps. Because we process pixels on the reference track sequentially, the potential matches satisfying the frame-level time constraint can also be quickly found by a sliding time window on the track. With this optimization, the overall complexity for finding the best pixel matching scheme is $O(|\boldsymbol{s}'_a| \cdot (\beta + \alpha^2))$, where β is the number of pixels that satisfy the frame-level time constraint. As α and β are typically small integers in actual experiments, the overall complexity is almost linear to the length of the reference track.

Chapter 6

3-D Spatial-Temporal Trajectory Reconstruction

Given the method to match two tracks from different views, stereo matching can be run on all pairs of tracks from different views. The next step is to fuse the matching information from all matched track pairs and reconstruct 3-D trajectories of motion with timestamps.

In this chapter, we first introduce the method to estimate subframe temporal offset among cameras, which lays a foundation for later multi-view fusing. Then, the method to reconstruct tracks with multi-view information from a single reference camera is presented. Next, the method to fuse information from different reference cameras is introduced. Finally, we present some optional post-processing algorithms.

6.1 Subframe Temporal Offset Estimation

To fuse multi-view information and get subframe temporal resolution, timestamps from different cameras have to be aligned by subframe level temporal offsets.

The subframe level temporal offset can be estimated from stereo matchings. This is because a correct stereo matching also indicates that the matched points correspond to a 3-D position of the point light source at a specific time. If we know the exact timestamps of the matched points in timelines of the two video clips respectively, the temporal difference between the two cameras is the difference of the two timestamps.

However, in reality, the temporal difference estimated from a single stereo matching pair may not be accurate because of the following reasons:

- Inaccurate timestamps. At this step, most timestamps of track points are estimated from timestamps of track endpoints. They may not be accurate enough for a direct temporal offset estimation.
- Each 2-D track point is from a pixel, which is not a sample of a point in the image plane, but an integration over the sensor's area. In that

sense, an accurate timestamp for a pixel does not even exist. It is always an approximation.

- The matching among points may not be one-to-one matching. When a point looks static in one view but moving in another view, the matching will be one-to-many or many-to-one. Estimations from such matchings cannot be all correct.
- The stereo matching itself might be incorrect.

To overcome the problems, we choose several stereo matchings that are likely to be accurate and average them to minimize the estimation error. One heuristic criterion is to choose matchings of pixels which are both from image streaks which are neither too short nor too long.

The result and overall accuracy of subframe temporal offset estimation is shown in Section 7.3.2.

6.2 Track Reconstruction with a Single Reference Camera

6.2.1 Two-View Reconstruction

For every two camera views, with each 2-D track as the reference track, one 3-D trajectory can be reconstructed from each of its matched tracks from the other camera. Ideally, these 3-D trajectories should be reconstructed from different parts of the reference track. However, practically the reconstructed tracks from one reference track may also conflict with each other, that is, two trajectories may cover some time span in common but have different 3-D locations. The conflict can be caused by two main reasons:

- A track is accidentally matched to the reference track by error. Such wrongly matched tracks are typically short.
- The assumption that one track corresponds to one physical light source is violated. If the reference track was actually left by two physical light sources with different trajectories, there can be two tracks from another view that can both match the reference track correctly.

To deal with the first source of conflict, a greedy strategy is applied. We model each point on the reference track as a slot, which needs to be filled by a corresponding 3-D point. Then we process the reconstructed tracks by their averaged per-pixel matching cost, in increasing order. Each time a reconstructed track is processed, we try to use the points on it to fill their corresponding slots in the reference track. If a slot has been filled, the current point is discarded.

During the slot filling procedure, we can test the validity of a matching by computing the percentage of points on a reconstructed track which are used to fill in slots. For example, if 50% of the points are discarded, the whole matching is probably "invalid". The threshold of validity check is flexible. Once a matching is determined to be invalid, the slots filled by it should be undone.

As the second source of conflict violates our basic assumption, we do not solve it at this stage. In a multi-view setup, the problem might be solvable in other views.

Finally, the 3-D points in the filled slots of a reference track form a 3-D trajectory (with missing points, which correspond to empty slots). Also, each pair of tracks from different cameras can be labelled with a boolean value, representing the validity of the matching.

6.2.2 Multi-View Fusing for Single Reference Track

In a multi-view setup, the two view matching method above can be done for every pair of views. As a result, for each 2-D track as the reference track, several 3-D trajectories can be obtained, each of which comes from a different camera view. According to our assumptions, all these tracks are actually from the same physical light source. Fusing these trajectories into one trajectory will help integrate information and improve the robustness of the whole system.

Because all the 3-D trajectories to be fused have the same 2-D reference track, fusing can be done by processing points on 3-D tracks sequentially according to their corresponding 2-D point on the reference track. Each time one 3-D point needs to be chosen from all reconstructed points with the same 2-D reference point.

Our basic strategy is to simply choose the median from all corresponding points from all the trajectories. It is easy and fast, and can effectively rule out many wrong matches.

If there are enough camera views (at least three), we also add a trajectory consistency check. For example, for each pixel on the reference track, at least two reconstructed trajectories should be consistent on the reconstruction result (the distance between the two reconstructed points should be within a certain threshold). If no consistency can be reached, the reconstructed point will be considered as an outlier and be discarded.

If a finer or more accurate fusing is desired, one may also consider taking into account how a trajectory point is reconstructed. For example, if the pixel used for reconstructing a 3-D point was chosen from a large pool of stereo matching candidates, the 3-D point is generally less reliable because there probably was a strong ambiguity and the selection was not based on the more reliable epipolar constraints. We have not implemented or tested this strategy.

After multi-view fusing, one 3-D trajectory is obtained from each reference 2-D track.

6.2.3 Temporal Super-Resolution

After a track is matched with tracks in other views, temporal information from other views can be integrated into the current view and temporal super-resolution can be done. For points reliably matched to points in other views with reliable timestamps, the matched point's timestamp can be used to label the points on the reference track so that the original track timestamps from linear interpolation can be replaced and improved.

Many problems in subframe temporal offset estimation occur in this step as well. For example, if a light source appears static in one view but moving in another, the matching will not be one-to-one and such matchings should not be used for temporal super-resolution.

Practically, it is possible that the newly labelled timestamps on the reference track are not increasing along the reference track, which implies that either matching errors have occurred or the simple trajectory assumption might be violated. In either case, the involved timestamps should be discarded and the track segment should be flagged as problematic so that it can be taken account in the multi-view fusing step.

Now, along each track, some points are labelled with some reliable strictly increasing timestamps. For the remaining points, the timestamps have to be interpolated.

Generally, to do timestamp interpolation, timestamps should be first parameterized by the accumulated arc length of the track:

$$g(s) = t,$$

where s is the accumulated arc length from one endpoint of the track[3] and t is the timestamp. In the case that the light source is static in part of the

[3]One can either use the length of the 2-D reference track or the length of the 3-D reconstructed track. The 2-D track is generally less noisy but lacks 3-D information, while the 3-D track includes 3-D information but can be noisy.

track, the function above will not be well defined. In such a case, one can either make small perturbations to s or divide the track into several track segments so that $g(s)$ is well defined in each segment.

Once the parameterization is done, interpolation can be done in different ways as long as the monotonicity of timestamps is preserved. For example, linear interpolation and monotone cubic interpolation are two of the possible choices.

The validity of such interpolation is backed by Newton's Laws of Motion, which ensures that the function of distance (or accumulated curve length) over time $s = f(t)$ should be continuously differentiable(C^1). For the track segments where the light source is not static, the inverse function theorem ensures that f's inverse function $f^{-1}(s) = g(s)$ exists, and it is also continuously differentiable. Many interpolation methods, such as monotone cubic interpolation, are actually finding a continuously differentiable function based on some general regularization principles and are thus valid.

It is worth noting that if one decides to smooth the track as post-processing, timestamp interpolation is better done after track smoothing so that the length parameter s will be less noisy.

6.3 Fusing Information from Different Reference Cameras

Although motion reconstruction can be done with only one camera as the reference camera, fusing information from multiple views can enhance the final motion reconstruction result in the following aspects:

- Some light sources might be missing in some of the views but visible in the rest. By fusing tracks reconstructed with different reference cameras, trajectories of light sources which are only visible in some of the views can be preserved.
- Some parts of a track might be reconstructed with low confidence (e.g., from matchings with high cost), replacing such parts with more confident tracks from other reference views may help correct some wrong matchings.
- A light source can move much slower in a particular view than in the others. According to our track matching method, the track reconstructed with the view as the reference view will have fewer point samples. With multi-view fusing, it is possible to use information from other views to enhance the spatial-temporal resolution of the track.

To effectively fuse multi-view information, reconstructed tracks from different reference cameras have to be grouped by the light source they correspond to and then fused, so that finally only one 3-D track is obtained from each light source. (If a light source disappears in all camera views and appears again, as there is no way for the motion capture system to know they are the same light source, we treat them as two different light sources.)

6.3.1 Grouping

Reconstructed 3-D tracks corresponding to the same physical light source need to be grouped first.

Grouping can be done using the information from pairwise stereo track matching, where two matched tracks are believed to correspond to the same light source. Specifically, the relation that two tracks belonging to one physical light source is an equivalence relation because it satisfies reflexivity, symmetry and transitivity. If all the tracks are viewed as a set, then tracks belonging to one light source are an equivalence class, which means that the 2-D track grouping problem is actually a set division problem. The set division problem can be solved precisely and efficiently by a disjoint set data structure and related algorithms. The time complexity is linear in the number of 2-D tracks plus the number of track matching pairs.

The algorithm above works well in most cases. However, it can fail when 2-D tracks of different light sources were matched in error. In such a case, all tracks of the two light sources will be grouped and the result will be incorrect. To avoid such problems, we take several measures:

- Put a strict threshold on the matching cost when determining if two tracks can be matched or not, so that only very confident track matchings are used in grouping. This might result in more discontinuous tracks, but it is much less likely to invalidate the whole result.

- Ignore matchings for very short tracks. Short tracks might just be matched by accidental errors.

- If tracks of more than one light source are grouped together despite the measures above, one can still detect the error by checking if some tracks with overlapping temporal coverages from one camera are grouped together. When such a problematic grouping does happen, one should try to solve the problem manually (by choosing a stricter threshold or by grouping the tracks manually) before continuing to the following steps.

The above measures turn out to be sufficient for our experiments. However, it remains an open problem to design an efficient algorithm to be robust to false track matchings and to be able to deal with streak overlapping.[4]

6.3.2 Fusing

Once tracks are grouped, the next step is to fuse each group of trajectories together.

Typically, a group has the following characteristics:

- Tracks in a group may be reconstructed from different stereo pairs. Particularly, the reference track can be different.
- Tracks in a group which have the same reference track cover different time durations and do not have timestamps in common.
- Tracks in a group which have different reference tracks may cover the same time duration although they may not be consistent with each other.

To fuse these tracks into one 3-D trajectory, we need to combine all these tracks and when there is a conflict, a decision should be made. To do this, we do a temporal scan on all the tracks. The procedure can be described as follows:

1. Align all tracks temporally according to the estimated subframe level temporal offsets.
2. Sort all tracks in the group by their starting time, in increasing order.
3. Start with the timestamp of the first point on the first track as the current time, and do a temporal scan:
   (a) From tracks reconstructed with each reference view, find the track point with the smallest timestamp greater than the current time. As a result, several candidate points from different reference views will be found.

[4]Modelling the problem as a graph problem might be a possible direction: one can build a graph with tracks as nodes and matchings between tracks as edges weighted by matching confidences. The grouping task can be rephrased as finding a minimum cut on the graph so that after removing the edges in the cut, tracks (nodes) from the same camera and with overlapping time durations will not be in the same connected component any more. We do not yet know if this problem can be solved in polynomial time.

   (b) Choose the best point among the candidate points, and append it to the final output track.
   (c) Advance the current timestamp to the timestamp of the chosen point.
   (d) Repeat this step until all the track points are processed.

An important problem of the procedure above is how to choose the best point in step 3(b). In our implementation, we do the following:

- If the timestamps of all the candidate points are greater than the "current time" by more than a frame, we do not choose any point and advance the current time to the smallest timestamp of all the candidate points directly.
- Otherwise, choose any point.

Our implementation works well for our experiments. But a more sophisticated choice strategy can probably lead to better results. We believe a better strategy should take the following factors into consideration:

- Points reconstructed from more confident matchings (e.g., less ambiguity and smaller matching cost) are preferred.
- Points reconstructed from points with faster on-image speed are preferred.

However, practically it is difficult because the candidate points do not have the same timestamps, and we are not able to compare points directly.

Another possible solution is to temporally resample the trajectories before fusing, so that during the temporal scan, candidate points will have the same timestamps and become comparable. But as it is hard to keep information about point level stereo matchings during resampling, it is still difficult to find a reasonable way to do the fusing. In this case, the best strategy to find the best track point after resampling we can suggest is to choose the median from the candidate points, which might work well if a large number of cameras are available.

6.4 Track Post-Processing

Optional post-processing can be done on raw reconstructed 3-D tracks to enhance the result or make the result better fit other applications. We will talk about two post-processing steps: smoothing and temporal resampling.

6.4.1 Smoothing

Although smoothness of the reconstructed 3-D trajectory has already been taken into consideration in stereo matching, the raw reconstructed trajectory may still be rough due to spatial quantization effects.

To do smoothing, the track has to be parameterized. Two variables can be used to parameterize the track: time and accumulated curve length. Each of them has pros and cons.

When parameterized by time, a track can be written as a function $f(t) = \boldsymbol{x}$, where $\boldsymbol{x}$ is the 3-D position of a point on the track. According to Newton's Laws of Motion, f should be continuously differentiable. Based on that, it is reasonable to fit a smooth curve to the raw 3-D track. However, the problem is that most of the timestamps on raw reconstructed tracks are estimated. Parameterizing a curve with inaccurate information may cause larger errors in the later steps.

Parameterizing a track by its arc length is a standard way for curve parameterization. The problem is that temporal information attached to each curve point is not utilized in such parameterization. For example, such a parameterization cannot distinguish between a continuously moving light source with a light source which stopped moving for some time and started moving later.

We choose the second option because although it does not utilize all the information available, it does not rely on unreliable information (estimated timestamps).

After parameterization, smoothing can be done in different ways. We smooth tracks by fitting a B-spline approximation of each 3-D track, which can be done by SciPy's function `splprep`. The method accepts a parameter, which controls how well the smoothed track should fit the original track.

6.4.2 Temporal Resampling

Until now, all points on a track are actually from pixels of the reference track. The points are not sampled equally in time. This can make our output incompatible with that of other motion capture systems.

One solution is to resample the track points equally in time, which can be done by the following procedure:

- Parameterize the curve by time $h(t) = \boldsymbol{x}$
- Find a way to interpolate points by time (e.g., linear interpolation or B-spline approximation).

- Take the new temporal sampling rate as input, determine the new time points and interpolate points by the new time points.

Chapter 7

Experiments

To demonstrate the proposed method, we build motion capture systems with consumer grade cameras, make a demo of human motion capture, and evaluate the accuracy of the system in different ways.

7.1 Implementation

7.1.1 Hardware Setup

Cameras

All our experiments are done with consumer level SONY camcorders (SONY HDR-XR500), which record interlaced video at 60 fields per second with a rolling shutter. After decoding and deinterlacing, videos of 60 frames per second at the resolution of 1920×1080 are produced.

Like most consumer level camcorders, the cameras can automatically adjust exposure time according to the lighting condition. In a common indoor scene or outdoor scene at night, the camera is usually in continuous exposure mode ($1/60s$ per scanline). This can be verified by the EXIF information in the video clip and the connectible streaks from adjacent video frames.

The cameras also have an setting for the "exposure" level, which affects the overall brightness of the video. To make marker segmentation easier, a proper exposure setting which dims the background and emphasizes the marker is preferred.

As static cameras are assumed throughout the system, to physically fix all the cameras during each capture, tripods, optical tables and super-clamps are used. There are no specific requirements on the position and orientation of the cameras, but good multi-view coverage of the experiment space is preferred.

Markers

As our motion capture system is designed for bright point light sources only, motion capture for general objects has to be done by attaching bright markers to them. Well-designed markers are important for a good segmentation and good reconstruction results.

The most critical factor on markers is their brightness. This is because pixels on streaks are only exposed by markers for a short portion of their exposure time and as a result the streaks can be blended into the background, which presents a big challenge to streak segmentation. Due to this reason, markers that work well in traditional short exposure setups may not be bright enough in our streak based system.

Considering the simplicity of passive markers, we first tested several types of them, including reflective tapes and retro-reflective sphere markers.

White reflective tapes turned out to be the worst choice because when their white streaks cannot saturate the sensors, they are hardly distinguishable from other white or gray objects (Fig. 7.1a). Reflective tapes with a distinguishable color in the capture environment (such as bright green, which is not common in a lab and quite different from the color of human skin) work better when color-based segmentation is done, but under fast motion the streaks can still be too dim to be segmented robustly (Fig. 7.1b).

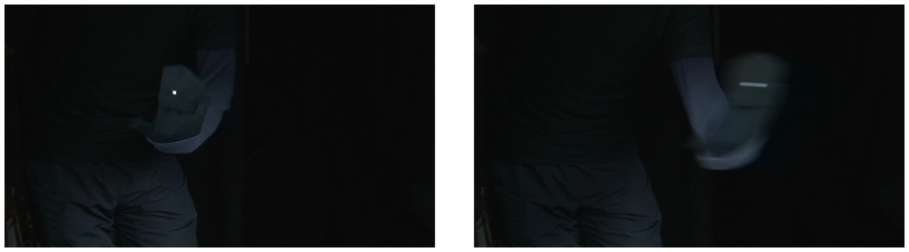
Retro-reflective white sphere markers used in Vicon motion capture system might work with directed lighting coming only from cameras, but in an environment without accurate lighting control, their streaks are not bright enough comparing to the background and can not be robustly segmented (Fig. 7.1c).

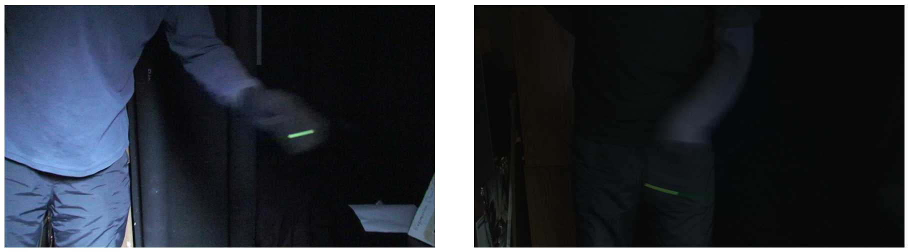
Besides, all these reflective markers require a dark scene and controllable light sources to work, making the whole setup difficult to achieve and less portable.

On the other hand, LED based active markers turn out to be more practical: even when driven by coin cell batteries, an LED can be much brighter than an indoor scene with mild illumination and can saturate camera sensors even under fast motion. When packaged with a proper lens, a wide angle of view can also be achieved so that it can approximately act as a point light source.

As no special modulation is needed on our LED markers, a simple circuit is enough to drive the LEDs. As shown in Fig. 7.2, the marker is small and light enough to be mounted on a human body or many other objects.

It is worth noting that because the current to drive an LED (5mA - 20mA) is significantly larger than a coin cell battery's standard continuous


(a) White reflective tape. Although it is bright enough when still, under fast motion it can blend with the background, making segmentation difficult.


(b) Green reflective tape. With a color based segmentation algorithm, it can work well when it is not too far from the light source and facing the cameras and light source directly. But for practical experiments, it is still far from being robust.

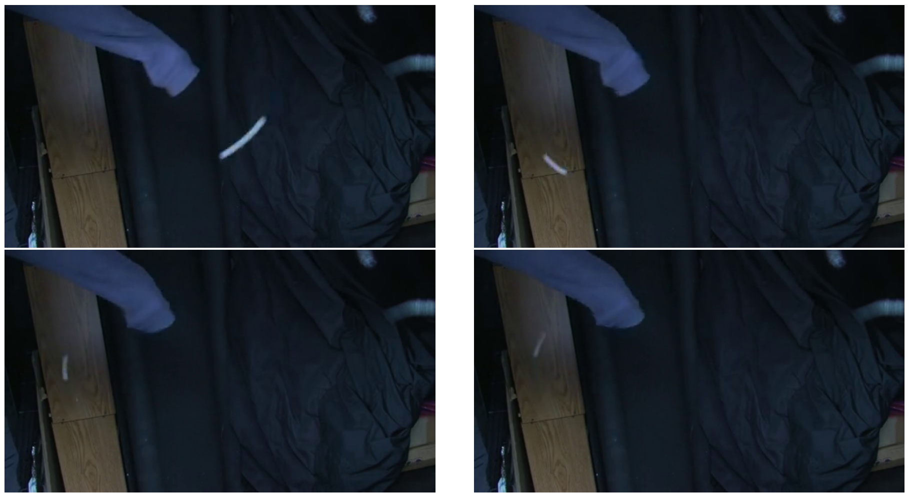
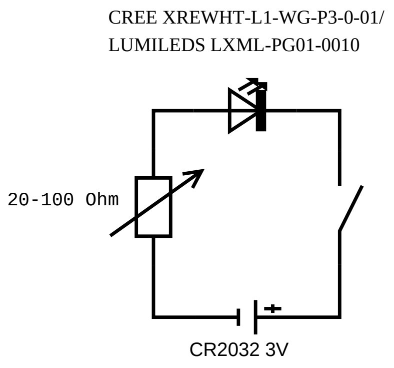

(c) Streaks of a retro-reflective marker from Vicon. The streaks can easily be too dim to be segmented robustly when the marker is moving fast or far from the light source. For such retro-reflective markers, a better directed light source might produce a better result, but it is hard to implement and test.

Figure 7.1: Challenges with passive reflective markers.



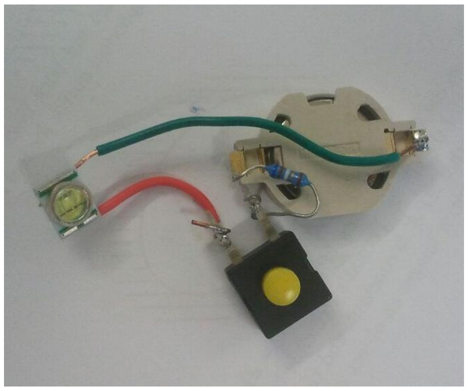


Figure 7.2: The circuit diagram for an LED Marker and a photo of an actual marker. The LED is driven by one CR-2032 3V coin cell battery and protected by a resistor. As a battery's internal resistance may grow while discharging, a potentiometer can be used to maintain constant brightness. However, a simple resistor as shown in the right figure is sufficient for our experiments.

discharge load (0.2mA), the internal resistance of a battery can be large (around $10 - 100\Omega$) and grow while discharging. A good and safe battery and a proper resistor are recommended for building a safe marker at required brightness.

Active LED markers proved to be the best for our experiments and we use them in all the following demos and experiments.

7.1.2 Software

As real-time data acquisition is not possible for the consumer grade cameras we use, we process video clips offline.

First, with `mplayer`, the video clips from cameras are deinterlaced with Yadif(2x) algorithm and converted into sequences of static images.

We implement most of our methods in Python with the help of scientific libraries (SciPy, OpenCV, Matplotlib, Mayavi 2) and the GUI library PyQt. To make the overall running time acceptable for doing experiments, some time consuming parts, such as triangulation and the Viterbi method, are implemented in Cython. Stereo matching of different view pairs is run in parallel to shorten running time.

Some steps are done through third party packages. Streak skeletonization and pruning is done using the Multi-Dimension Array (MDA) package; calibration is done using the open source Multi-View Self-Calibration Matlab code from Svoboda et al. [33].

The processing of the system can be divided into three stages, track extraction, stereo matching and multi-view fusing. On an Intel i7-2600K CPU at 3.40GHz, the track extraction step takes about 0.2 second per frame per marker (depending on the thickness of the streaks as iterative thinning is done). For the stereo matching step, the average CPU time to process one frame per stereo pair is typically much less than a second (depending on the lengths of tracks). The multi-view fusing step is relatively fast (about several milliseconds per frame).

The relatively slow performance is related to the Python programming language and the NumPy library. According to the complexity analysis, we believe that with a careful implementation in a lower level language, real-time computing (with a small delay) of stereo matching and multi-view fusing should be possible.

7.2 Demos

7.2.1 Human Motion Capture

Human motion capture is a very popular application scenario for motion capture. We built a proof of concept human motion capture system which is able to capture fast motion with slow cameras.

In our demo, due to hardware limitations, a simplified human skeleton model (shown in Fig. 7.3) of 15 landmark points is used. Correspondingly, 15 battery powered LEDs are attached to a human body by Velcro and tape (Fig. 7.3).

To have a good coverage of the experiment space, we used a 7 camera setup as shown in Fig. 7.4. The cameras are controlled by remotes and touch screens.

After camera calibration (as described in Section 3.5), an arbitrary 3-D Euclidean coordinate system is established. To convert this coordinate system to the room coordinate system (with its $X - Y$ plane as the floor), another calibration step is needed. To do the calibration, a 3-LED calibration wand from Organic Motions is used. The 3 LEDs on the wand form a right triangle. When the wand is put on the floor of the experiment space, the cameras will see the LEDs and are able to reconstruct the 3-D positions of the LEDS in the current coordinate system. The transformation matrix between the two coordinate systems can then be determined.

To get the stick figure visualization, the reconstructed 3-D motion trajectories are manually labelled with the LED it corresponds to. To make the labelling process easier, a graphical user interface is used so that one can see the reprojected trajectory in original camera views when labelling a 3-D trajectory.

We captured several different types of motion, including walking, jumping, running on the spot and kicking. Slow motion videos of the captured motion are provided as supplemental materials.

7.2.2 Spinning Fan with a Rotating Head

To demonstrate the system's ability to deal with fast motion, the motion of a blade of a spinning cooling fan with a rotating head is captured and reconstructed.

A battery powered LED is attached to a blade of the cooling fan, and five cameras are used to capture the motion of the LED.

Once stable, the spinning speed of the fan is around 1000rpm, which implies that a streak exposed in 1/60 second will cover almost 1/3 of a

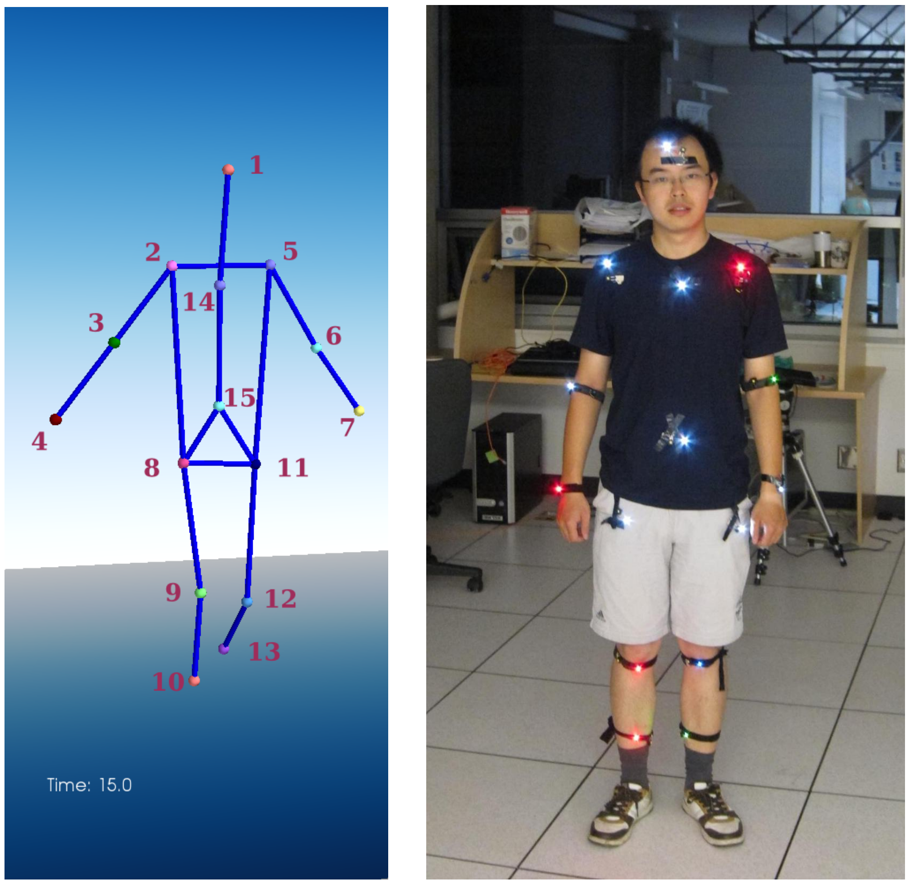


Figure 7.3: The simplified human skeleton of 15 landmark points. A photo of the actual LED mounting positions is shown on the right.

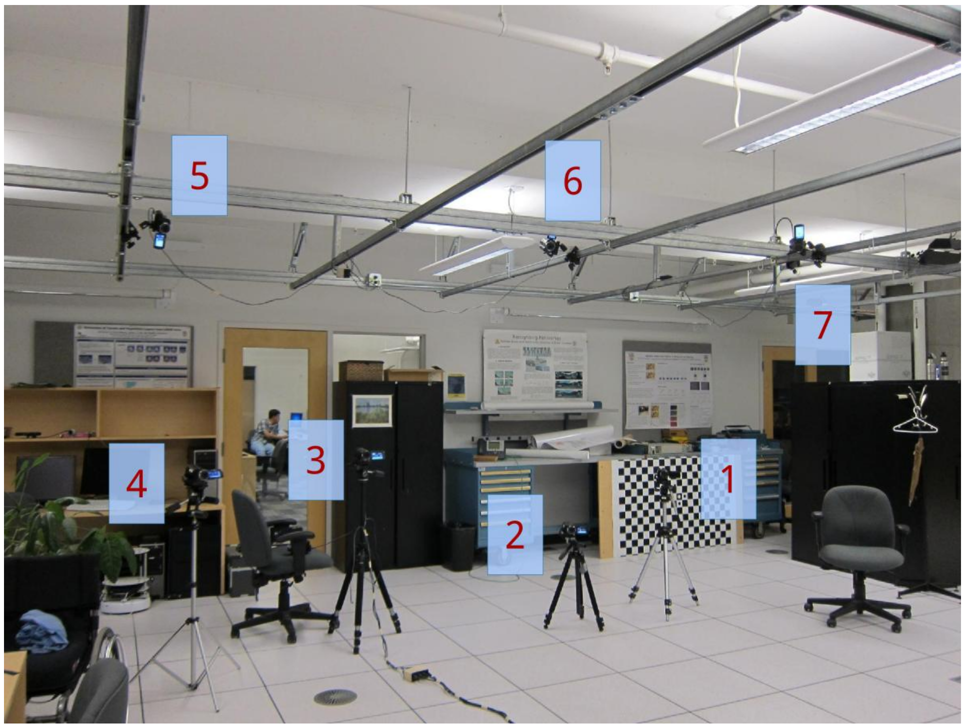


Figure 7.4: A seven-camera setup for a human motion capture demo. The cameras on the ceiling and near the floor can help reduce occlusion and multi-view stereo ambiguity.

circle. Sample frames from the acquired video are shown in Fig. 7.5.

The reconstructed trajectory is shown in Fig. 7.6.

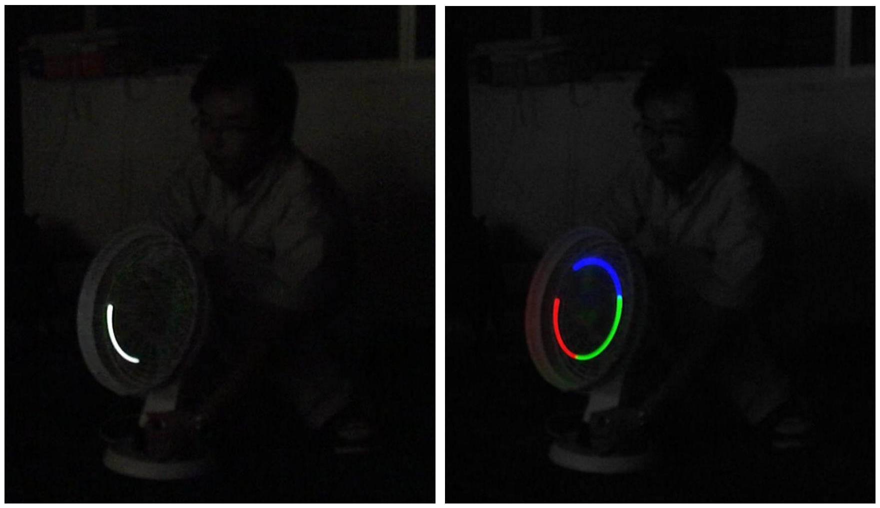


Figure 7.5: Frames from the spinning fan video clip. The left image is a single frame, while the right image shows three consecutive frames in its blue, green and red channel, respectively. The fan's spinning speed is not constant, but its trajectory is roughly a circle.

7.3 Evaluation

It is important to validate the correctness of the result and evaluate the accuracy of the proposed system. However, we neither generate fully predictable motion nor provide an independent method to measure motion with high temporal and spatial accuracy, we cannot evaluate our system directly. Instead, we evaluate the overall accuracy indirectly through spatial accuracy and rigidity.

Many steps of our system can introduce errors. Ideally, we would like to test the correctness and accuracy of every step, but due to practical difficulties to get ground truth data for each individual step, we have to concentrate on evaluating the overall result. Only two steps in the system, geometric calibration and subframe level temporal synchronization, are evaluated in dedicated experiments.

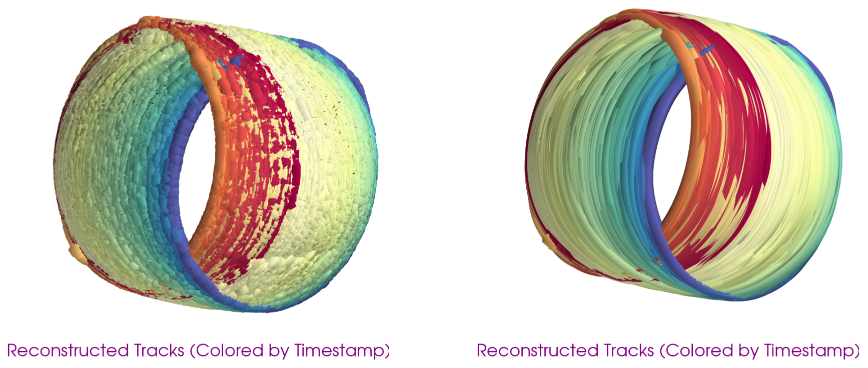

Reconstructed Tracks (Colored by Timestamp)
Reconstructed Tracks (Colored by Timestamp)

Figure 7.6: Reconstructed 3-D trajectory for an LED on a blade of a spinning fan with its head rotating, before and after smoothing. The points on the trajectories are colored by their timestamps. Jitters in the left figure are probably from the quantization effect. But jumps in some parts of the trajectory indicate that errors are still present in the matching result.

7.3.1 Geometric Calibration

We calibrate cameras using a blinking LED and the toolbox from [33], which takes in synchronized video frames as input and does LED localization and camera calibration internally. The blinking LED synchronization step works very well in our experiments. So the accuracy of geometric calibration is mainly determined by the toolbox we use and the quality of data.

Although the accuracy is not directly related to our work, it can largely affect the stereo matching and overall accuracy of the final result. Getting an idea of the geometric calibration accuracy can help us better understand other evaluation results.

The experiment was done with 5 cameras and 1173 LED blinks (not all the blinks were visible to all the cameras). The result is shown in Table 7.1 and Fig. 7.7.

Note that although the average reprojection errors are small, the number of inliers is not as large as expected. Besides, a large variation is shown among the intrinsic camera parameters of different cameras, while the cameras are actually of the same model and were under the same settings.

The problems in calibration are probably caused by the following factors:

- The optimization algorithm of the calibration toolbox optimizes the 3×4 projection matrix as a whole. Constraints and prior knowledge of the intrinsic camera parameters cannot be applied during optimization.
- Estimating the position of an LED in an image by the centroid of the segmented blob is not always accurate. This might also be one reason for the small number of inliers.
- The LED positions do not always have full coverage of all the camera views.

As we have little control over these factors, we decide to continue with the presence of these errors.

7.3.2 Subframe Temporal Offset Estimation

To validate the stereo matching based subframe temporal offset estimation, we do the estimation together with strobe based subframe temporal offset measurement (described in Section 3.4.2) in one capture, so that the estimation can be evaluated by the measurement.

In the experiment, we find that the standard deviation of the per point temporal offset estimations is very small (typically 0.03-0.06 frame duration,

	LEDs	Inliers	Principal Point	F_x, F_y
Camera #0	423	336	(1025.9, 515.9)	2080.6, 2082.0
Camera #1	508	398	(928.2, 645.8)	2104.0, 2098.2
Camera #2	585	497	(1004.4, 459.7)	2048.9, 2047.2
Camera #3	587	497	(940.4, 518.0)	2010.6, 2013.6
Camera #4	410	335	(944.5, 560.5)	2041.6, 2041.9

Table 7.1: A calibration result showing the problems. The second column shows the number of frames found to contain a full clear LED, while the third column shows the number of inlier LED positions under the estimated calibration. The estimated principal point (the resolution of a frame is 1920×1080) and focal length in horizontal and vertical direction are shown in the fourth and fifth column. The data show a large variation among cameras, but the cameras are actually all of the same model and were under the same settings.

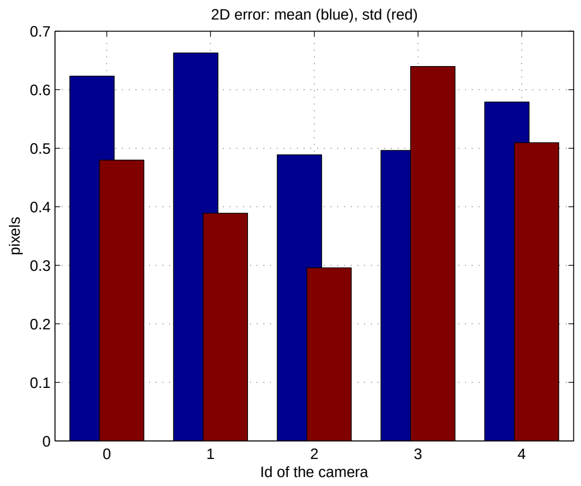


Figure 7.7: Reprojection error of the calibration result.

or 0.5-1ms) for points from long streaks. An example of the distribution of the estimated temporal offset per pixel is shown in Fig. 7.8, together with the result from the strobe based subframe temporal offset measurement.

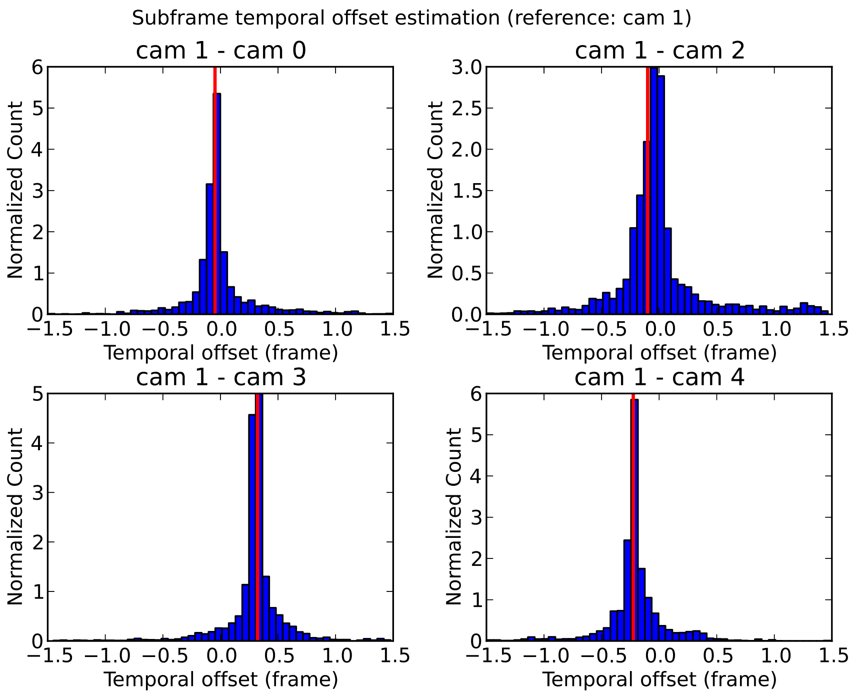


Figure 7.8: Subframe temporal offset estimation by stereo matching. The red lines are the temporal offset measured by the strobe based method (Section 3.4.2), which is reliable and can act as the ground truth. The distribution of the difference of timestamps between matched pixels is shown as the blue histograms. The estimation is highly consistent with the measurement.

7.3.3 Spatial Accuracy

Although it is hard to know both the spatial and temporal information for motion exactly, the trajectories of some motions are known. For such motions, comparing the reconstructed trajectory to the ground truth trajectory can provide an evaluation of the overall spatial correctness and accuracy.

When the motion is fast, evaluation on spatial correctness and accuracy also has implications in the temporal domain. As a correct 3-D spatial trajectory has to be reconstructed from correct stereo matchings, the correctness of stereo matching and reconstruction can also indirectly reflect the correctness in the time domain qualitatively.

Setup

In our experiment, motion with known trajectory is generated by the same spinning fan in Section 7.2.2 (but with its head fixed), on which the trajectory of any point should always be a planar circle.[5] A photo of the setup is shown in Fig. 7.9.

Evaluation

The reconstructed 3-D trajectory of the LED before and after post-processing is shown in Fig. 7.10, which can give an intuitive impression on the result.

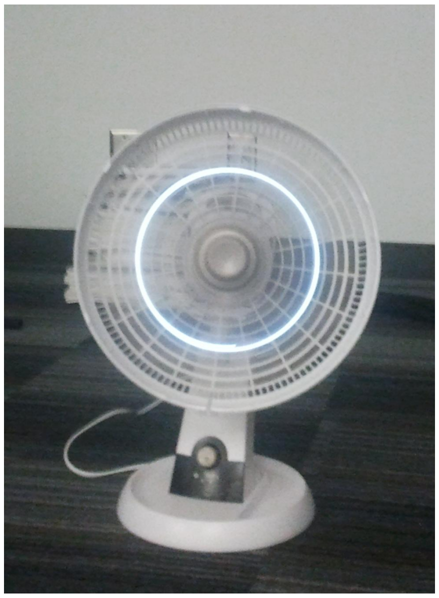


Figure 7.9: The spinning fan photoed by a mobile phone with a 1/8 second exposure. Note that the streak is not a perfect ellipse because the fan has visible jitter.

[5]Because an LED with battery has to be attached to a blade of the fan, the fan does not have constant spinning speed any more and as a result we are not able to know the temporal ground truth to do a direct evaluation on the temporal accuracy.

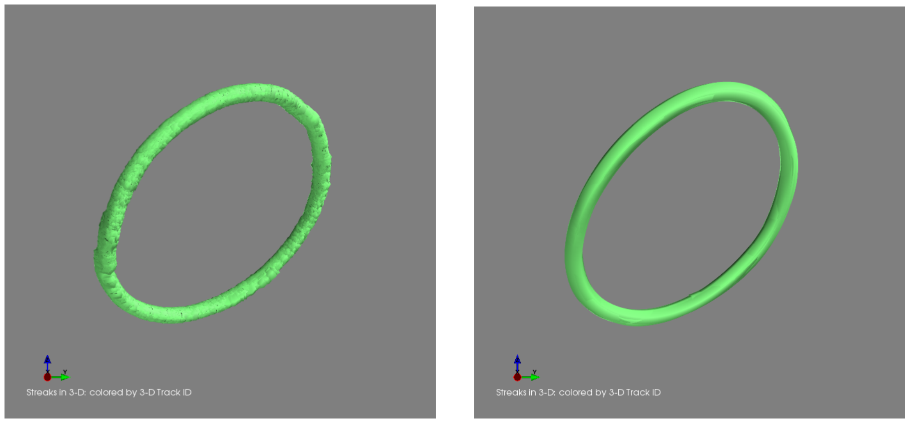

Streaks in 3-D: colored by 3-D Track ID
Streaks in 3-D: colored by 3-D Track ID

Figure 7.10: Reconstructed 3-D trajectory for the LED on the spinning fan, before and after smoothing. Jitters in the left figure are probably from quantization. But jumps in some parts of the trajectory indicate that errors are still present in the matching result.

As it is hard to accurately measure the absolute position of the fan and the LED, quantitative evaluation is done by testing how well the trajectory fits a circle.

For a video clip of 200 frames (3.3 seconds), 34384 3-D points are reconstructed and form about 60 cycles of spinning motion. A 2-D circle is fitted using all points with the method shown in Appendix B. The distance from points to the fitted circle center is 13.59 ± 0.14cm (the standard deviation should be 0 for a perfect result). The distance from the points to the fitted circle plane is 0.30 ± 0.18cm (both the distance and standard deviation should be 0 for a perfect result).

After smoothing, the distance from points to the fitted circle center is 13.59 ± 0.13cm, and the distance from the points to the fitted circle plane is 0.29 ± 0.18cm.

Considering the limited accuracy in camera calibration and the jitter of the fan itself (Fig. 7.9), this result is acceptable. However, the result also demonstrates that we still need a good method to identify and remove points from wrong matches in the trajectory.

7.3.4 Rigidity Test

The relative position of points on a rigid object should remain the same no matter how the object moves. For a fast moving object, evaluating its rigidity can indirectly reflect both temporal and spatial accuracy of the system.

Setup

As shown in Fig. 7.11, three LEDs are installed on an rigid object, forming a right triangle. We test rigidity by checking the relationship of the reconstructed positions of the three LEDs.

Results

Based on 500 measurements of the waving wand, the lengths of the two legs of the right triangle are 22.68 ± 0.95cm and 49.04 ± 1.32cm, respectively. The angle between the two legs is 89.42 ± 2.84 degrees.

7.3.5 Discussion

Every step of our method can contribute to the observed errors. Specifically, geometric calibration, temporal offset measurement, streak segmenta-

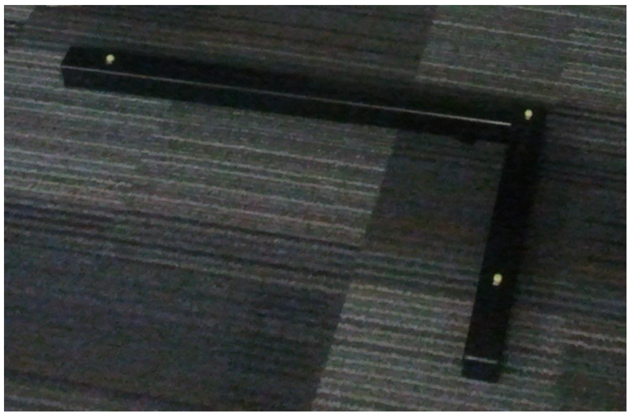


Figure 7.11: The 3-LED wand for rigidity test. The wand is from Organic Motion's calibration system.

tion, track extraction, stereo matching, stereo reconstruction and multi-view fusing may all cause errors.

Generally, although quantitative evaluations of the overall accuracy of our system are done, we are still not able to analysis how much an individual step contributes to the overall error at this point. If people are interested in getting such information, we recommend doing experiments on synthesized data sets, so that the impact of errors in each individual step can be analyzed.

Chapter 8

Conclusion and Future Work

8.1 Limitations

Although our system has advantages over traditional optical motion capture in several aspects, inevitably it also has limitations due to both theoretical and practical reasons.

8.1.1 Sampling Rate

As all other digital systems, the streak-based motion capture system is limited by the sampling rate, both spatially and temporally. Spatially, the system is not aware of motion at a sub-pixel level. Temporally, we are limited by the timestamps we get from the cameras.

Further, as we are not sampling at time points but doing a long exposure, we have additional requirements to the motion: an object's trajectory should not be self-intersecting within the duration of a frame. For general curve intersections, our system can usually detect the problem and discard those frames. But for some special cases, such as a light source moving back and forth along its trajectory within a frame duration, the system is not able to detect the problem at all. The root cause for such problems is the insufficient temporal sampling rate with respect to the motion.

8.1.2 Intersecting and Overlapping Streaks

Practically, in a frame, two streaks might intersect with each other, or even be overlapping, which presents a challenge to track extraction and marker identification.

Our implementation is not able to separate intersecting streaks from a frame and discards all of them. As a result, if an intersection is caused by an intersection of the 3-D trajectories, it will occur in all the camera views and the motion trajectory near the intersection will be discarded. However, if the intersection is caused by projection, it is still possible that the trajectory can be recovered from other camera views without streak intersection.

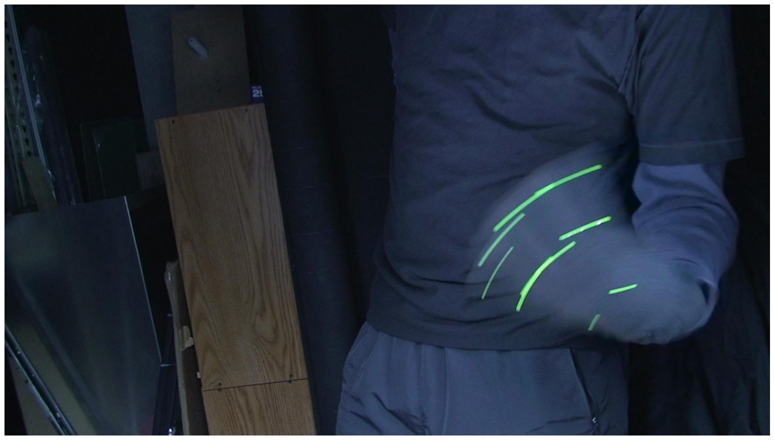


Figure 8.1: A frame with severe streak intersection and overlap. It is impossible to separate the streaks of the 10 markers from the frame.

Streak overlapping is a different problem. For our implementation, the main outcome is that tracks of different light sources will be incorrectly merged. It does not matter if the 3-D trajectories of the light sources are actually overlapping or not.

These limitations are specific to our method and implementation. As discussed in Section 4.2.4 and Section 6.3.1, a more sophisticated method may solve these problems with proper assumptions.

8.1.3 Hardware and Environment

Similar to other marker based optical motion capture systems, our system requires the light sources (or markers in a marker based setup) to be clearly visible to the cameras. Further, we have an additional requirement that the streaks of the light sources must be of high quality, which implies that the streaks should be clear enough so that segmentation can be done robustly and the medial axis of a streak should represent the trajectory of the center of the light source.

Besides, as the video cameras have to be in continuous exposure mode, the environment must not be too bright.

The requirements are stronger than traditional optical motion capture systems, but with bright LEDs as markers, the practical difference is not too large.

8.2 Conclusion

In conclusion, we present a streak-based motion capture system with consumer grade unsynchronized cameras, which is much easier to set up and much cheaper than current commercial alternatives. With the proposed methods to do camera synchronization and calibration, streak segmentation and analysis, stereo matching and multi-view information fusing, the system can capture and reconstruct the full 3-D trajectory of fast motion with high temporal resolution as long as our assumptions are met. The correctness and effectiveness of the system are shown by demos and experiments on capturing human motion and spinning fan motion.

Future work can be done in several directions: (1) improve the method to make it work with fewer assumptions and run faster; (2) test the system thoroughly to see how much the errors of each step can affect the overall result; (3) build applications for the system.

Specifically, to improve the system itself, some possible directions are:

- A cleverer segmentation algorithm which takes shape information into account, so that the light sources do not have to be extremely bright in the frames.
- An efficient way to take multi-view information into account while doing stereo matching.
- Make the system work with intersecting and overlapping streaks (see discussions in Section 4.2.4 and Section 6.3.1).
- A better multi-view fusing algorithm so that multi-view information can be fully utilized (as discussed in Section 6.3.2).

Bibliography

[1] Bradley Atcheson, Felix Heide, and Wolfgang Heidrich. CALTag: High Precision Fiducial Markers for Camera Calibration. In Reinhard Koch, Andreas Kolb, and Christof Rezk-Salama, editors, *VMV*, pages 41–48. Eurographics Association, 2010.

[2] Bouguet, Jean-Yves. Camera Calibration Toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`. Accessed: 31/08/2012.

[3] D. Bradley, B. Atcheson, I. Ihrke, and W. Heidrich. Synchronization and rolling shutter compensation for consumer video camera arrays. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 1–8, june 2009.

[4] R. Cavallaro. The FoxTrax hockey puck tracking system. *Computer Graphics and Applications, IEEE*, 17(2):6–12, 1997.

[5] W.G. Chen, N. Nandhakumar, and W.N. Martin. Image motion estimation from motion smear-a new computational model. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(4):412–425, 1996.

[6] Brian Clegg. *The Man who Stopped Time: The Illuminating Story of Eadweard Muybridge : Pioneer Photographer, Father of the Motion Picture, Murderer*, pages 123–126. Joseph Henry Press, May 2007.

[7] S. Dyer, J. Martin, and J. Zulauf. Motion capture white paper, 1995.

[8] ElphelWiki. Camera Synchronization - ElphelWiki. `http://wiki.elphel.com/index.php?title=Camera_Synchronization`. Accessed: 28/08/2012.

[9] J.S. Fox. Range from translational motion blurring. In *Computer Vision and Pattern Recognition, 1988. Proceedings CVPR'88., Computer Society Conference on*, pages 360–365. IEEE, 1988.

[10] Peter Fuller. Some Highlights in the History of High-Speed Photography and Photonics as Applied to Ballistics. In Lalit Chhabildas, Lee Davison, and Yasuyuki Horie, editors, *High-Pressure Shock Compression of Solids VIII*, Shock Wave and High Pressure Phenomena.

[11] M. Furniss. Motion capture. In *MEDIA IN TRANSACTION CONFERENCE*, volume 1, 1999.

[12] Michael Gleicher. Animation from observation: Motion capture and motion editing. *SIGGRAPH Comput. Graph.*, 33(4):51–54, November 1999.

[13] Rafael C Gonzalez and Richard E Woods. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J., 3 edition, 2008.

[14] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[15] Jee-Hoon Jung and Sung-Hoon Hong. Deinterlacing method based on edge direction refinement using weighted maximum frequent filter. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '11, pages 36:1–36:5, New York, NY, USA, 2011. ACM.

[16] Michael Koch, Zoran Zivkovic, Richard Kleihorst, and Henk Corporaal. Distributed smart camera calibration using blinking LED. In *Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems*, ACIVS '08, pages 242–253, Berlin, Heidelberg, 2008. Springer-Verlag.

[17] George Lassanske. Photographing high speed motion. *The Wisconsin engineer*, 63(2):22–27, November 1958.

[18] C. Lei and Y.H. Yang. Tri-focal tensor-based multiple video synchronization with subframe optimization. *Image Processing, IEEE Transactions on*, 15(9):2473–2480, 2006.

[19] C.K. Liang, L.W. Chang, and H.H. Chen. Analysis and compensation of rolling shutter effect. *Image Processing, IEEE Transactions on*, 17(8):1323–1330, 2008.

[20] Christian Linz, Timo Stich, and Marcus Magnor. High-speed Motion Analysis with Multi-exposure Images. In *Proc. Vision, Modeling and Visualization (VMV) 2008*, October 2008.

[21] Edwin George Lutz and Cornell University Library. *Animated cartoons; how they are made, their origin and development.* New York, C. Scribner's sons, 1920.

[22] Alberto Menache. *Understanding Motion Capture for Computer Animation and Video Games.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1999.

[23] T.B. Moeslund. Computer vision-based human motion capture–a survey. *University of Aalborg Technical Report LIA*, 99(02), 1999.

[24] T.B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.

[25] E. Muybridge, Sterling, and Francine Clark Art Institute. *Animal locomotion.*

[26] S.K. Nayar and M. Ben-Ezra. Motion-based motion deblurring. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(6):689–698, 2004.

[27] R.H. Nixon, S.E. Kemeny, B. Pain, C.O. Staller, and E.R. Fossum. 256$\times$ 256 cmos active pixel sensor camera-on-a-chip. *Solid-State Circuits, IEEE Journal of*, 31(12):2046–2050, 1996.

[28] Organic Motion Inc. Organic Motion: Markerless Motion Capture. `http://www.organicmotion.com/`. Accessed: 23/08/2012.

[29] Kalman Palagyi. Skeletonization techniques. `http://www.inf.u-szeged.hu/~palagyi/skel/skel.html`. Accessed: 13/08/2012.

[30] L. Rabiner and B. Juang. An introduction to hidden Markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[31] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, volume 2, page 7, 2011.

[32] P. Shrestha, M. Barbieri, H. Weda, and D. Sekulovski. Synchronization of multiple camera videos using audio-visual features. *Multimedia, IEEE Transactions on*, 12(1):79–92, 2010.

[33] Tomáš Svoboda, Daniel Martinec, and Tomáš Pajdla. A Convenient Multi-Camera Self-Calibration for Virtual Environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, August 2005.

[34] Christian Theobalt, Irene Albrecht, Jörg Haber, Marcus Magnor, and Hans-Peter Seidel. Pitching a baseball: tracking high-speed motion with multi-exposure images. *ACM Trans. Graph.*, 23(3):540–547, August 2004.

[35] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustmenta modern synthesis. *Vision algorithms: theory and practice*, pages 153–177, 2000.

[36] Andreas Velten, Everett Lawson, Andrew Bardagjy, Moungi Bawendi, and Ramesh Raskar. Slow art with a trillion frames per second camera. In *ACM SIGGRAPH 2011 Posters*, SIGGRAPH '11, pages 13:1–13:1, New York, NY, USA, 2011. ACM.

[37] Vicon Motion Systems. Motion Capture Systems from Vicon. `http://www.vicon.com/`. Accessed: 23/08/2012.

[38] B. Wilburn, N. Joshi, V. Vaish, M. Levoy, and M. Horowitz. High-speed videography using a dense camera array. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–294. IEEE, 2004.

[39] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 666 –673 vol.1, 1999.

[40] Zhengyou Zhang. Microsoft Kinect sensor and its effect. *MultiMedia, IEEE*, 19(2):4 –10, feb. 2012.

Appendix A

Blinking LED for Synchronization

To physically implement a blinking LED with controllable frequency, 555NE, a timer IC capable of generating square wave signal, is used to drive the LED. As shown in Fig. A.1, R_1 and R_2 together control the frequency of the LED, while their ratio decides the duty cycle. Practically, R_1 and R_2 can be two potentiometers so that timing can be flexibly controlled to adapt to different lighting conditions. Two CR2032 coin cells turned out to be enough to power the whole circuit. The whole setup is portable.

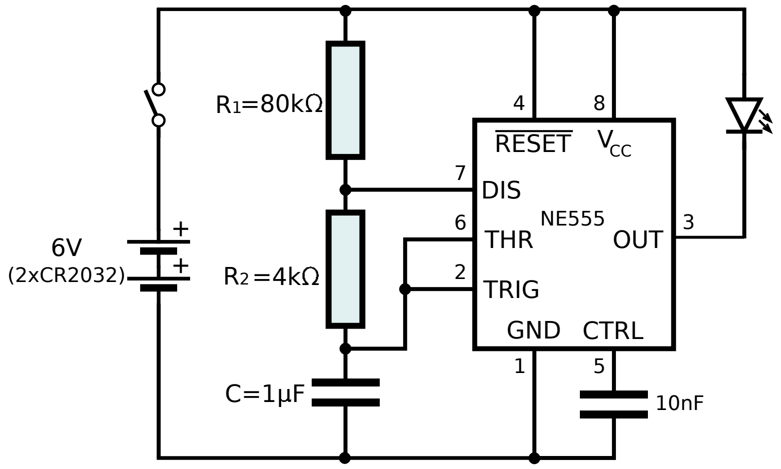


Figure A.1: The circuit diagram for a blinking LED that can be used for camera calibration. The short pulse of the LED optically synchronizes the cameras. (Figure derived from Wikipedia)

Appendix B

Fitting A Circle

Assume a set of 3-D points X is given, the centre of the circle can be estimated by:

$$C = \frac{1}{|\boldsymbol{X}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}} \boldsymbol{x}$$

The radius of the circle can be estimated by

$$r = \frac{1}{|\boldsymbol{X}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}} ||\boldsymbol{x}||_2$$

Then shift the trajectory to the centre accordingly:

$$\boldsymbol{X}' := \{\boldsymbol{x} - C | \boldsymbol{x} \in \boldsymbol{X}\}; \quad \boldsymbol{x}' := \boldsymbol{x} - C$$

To fit these points on a plane, do singular value decomposition on X',

$$\boldsymbol{X}' = U \Sigma V^T$$

The normal vector of the fitted plane is then the last column of V:

$$\boldsymbol{n} = V_{\cdot,3}$$

The distance from a point $\boldsymbol{x}$ to the fitted plane is then $|\boldsymbol{x}' \cdot \boldsymbol{n}|$, and the angle between $\boldsymbol{x}'$ and $\boldsymbol{n}$ is:

$$\cos^{-1} \frac{\boldsymbol{x}' \cdot \boldsymbol{n}}{||\boldsymbol{x}'||_2}$$

Finally, to tell how well the points fit the circle, the distribution of distances between track points and the fitted circle centre $||\boldsymbol{x}'||_2$, and the distribution of angles between every vector $\boldsymbol{x}'$ and $\boldsymbol{n}$, should be investigated.