Tracking the Joints of Articulated Objects
Without an *a priori* Shape Model

by

Simon Léonard

B.Eng. École de technologie supérieure, 1999

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming

to the required standard

The University of British Columbia

April 2002

© Simon Léonard, 2002
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
Vancouver, Canada

Date 2002/04/11
Abstract

The conventional methodology for tracking articulated objects relies on the knowledge of a shape model of the tracked object. The disadvantage of these top-down methods is their dependence on many assumptions, controlled environment and a priori knowledge. To be able to track and analyze arbitrary articulated motion, a radically different approach is necessary. This thesis proposes a method that addresses the lack of generality and adaptability of previous tracking systems for articulated objects. The method is built around an appropriate generic representation of articulated objects, which is a simple structural representation based on the joints configuration. The introduced method is a bottom-up approach able of detecting the joints of a moving object without any knowledge or assumptions. The method starts by extracting the moving contours of the object. Then the Hausdorff distance is used to decompose the contours into rigid components. From this decomposition, the joints are found by detecting the intersecting edge segments between adjacent components. Finally, the joints are then tracked by using an ad hoc deterministic version of the CONDENSATION algorithm.
Contents

Abstract ii

Contents iii

List of Figures v

Acknowledgements vii

Dedication viii

1 Introduction 1

2 Previous Work 4

2.1 Tracking With a priori Knowledge ............... 5
  2.1.1 3D Shape Models ............................ 5
  2.1.2 2D Models .................................. 6
  2.1.3 Stick Figures ................................ 7
  2.1.4 Others .................................... 7

2.2 Tracking Without a priori Knowledge ............ 8

3 Identification vs. Categorization 10

3.1 Identification .................................... 10

3.2 Categorization .................................. 11
  3.2.1 Structural Decomposition ...................... 11
  3.2.2 Geometric Constraints ....................... 12
  3.2.3 Multidimensional Feature Spaces ............ 12
  3.2.4 Second Order Isomorphism ................... 12
  3.2.5 Shape Models ................................ 13
  3.2.6 Mental Models ............................... 13
List of Figures

4.1 Intersection of two regions enclosed by deformable models. . . . 17
4.2 Piecewise decomposition of an articulated object. . . . . . . 17
4.3 Box diagram of the tracking system. . . . . . . . . . . . . . . 18
4.4 Joint detection. . . . . . . . . . . . . . . . . . . . . . . . . . . 19

5.1 Combined interframe difference of three frames. . . . . . . 22
5.2 High order moment filtering removes Gaussian noise. . . . . 23
5.3 Detection of moving edges. . . . . . . . . . . . . . . . . . . . 24

6.1 The Hausdorff distance on two sets of points. . . . . . . . 27
6.2 Robustness to clutter of the Hausdorff distance. . . . . . . 28
6.3 Partial Hausdorff distance violates the triangle inequality. . 31
6.4 The box reverse distance. . . . . . . . . . . . . . . . . . . . . 32

7.1 Voronoi diagram. . . . . . . . . . . . . . . . . . . . . . . . . . . 35
7.2 Masks for the chamfer algorithm. . . . . . . . . . . . . . . . 35

8.1 Algorithm for the detection of articulation. . . . . . . . . . . 40
8.2 Two sets of observations are made for each frame. . . . . . 42
8.3 Particle sampling: An approximation of a density by particles. 43
8.4 The CONDENSATION algorithm. . . . . . . . . . . . . . . . 44
8.5 Particles propagation. . . . . . . . . . . . . . . . . . . . . . 46

9.1 Hand data. . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
9.2 Hand joint detection. . . . . . . . . . . . . . . . . . . . . . 51
9.3 Gait data from [43]. . . . . . . . . . . . . . . . . . . . . . 52
9.4 Gait joint detection. . . . . . . . . . . . . . . . . . . . . . 54
9.5 Moving right arm data. . . . . . . . . . . . . . . . . . . . . 55
9.6 Moving left arm data. . . . . . . . . . . . . . . . . . . . . . 56
9.7 Moving both arms data. .............................................. 57
9.8 Forearm decomposition. ........................................... 59
9.9 Upper arm decomposition. ....................................... 60
9.10 Elbow detection. .................................................. 60
9.11 Right elbow detection. .......................................... 61
9.12 Left elbow detection. ............................................ 62
9.13 Both elbows detection. .......................................... 63
9.14 Both elbows detection. .......................................... 64
9.15 Right elbow tracking. ............................................ 65
9.16 Left elbow tracking. ............................................. 66
9.17 Both elbows tracking. ............................................ 67

10.1 Relation between coordinate systems. ......................... 69
Acknowledgements

This research has been fully supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) PGSA-221449-1999.

SIMON LÉONARD

The University of British Columbia
April 2002
Labor omnia vincit.
Section 1

Introduction

Tracking articulated objects and in particular the human body is an area of computer vision that has experienced soaring popularity in recent years. This interest is mainly generated by applications in the areas of human-computer interaction and gesture recognition [42, 8] whose thirst for systems that are able to understand visual commands (arms and hands) seems unquenchable. Other related areas include gesture analysis [64], surveillance and monitoring systems [15] and gait analysis [43, 53].

Perhaps the major difficulty with tracking articulated objects is their high degree of freedom. Unlike structure from rigid motion [61, 59] that involves six degrees of freedom, articulated objects imply a higher dimensionality. This results in a huge search space from which the pose of the object must be determined. For a complex object like the human body, models with up to 29 degrees of freedom can be used [14].

Because of this high dimensionality, researchers have resorted to different methods other than classic structure from motion techniques. With only a few exceptions, most of these methods require a priori knowledge about the object in order to determine the pose. For most cases, this is a reasonable assumption since most of these systems are developed for specific applications. As such, researchers often prioritize performance and accuracy over generality and adaptability. However, when experimenting outside specific problems and environment, tracking arbitrary articulated objects requires a radically different approach.

As mentioned, fewer attempts were made in trying to create a system for tracking articulated objects that is: fully autonomous and adaptable, capable of tracking and analyzing arbitrary articulated motion simultaneously and...
performing outside a controlled environment. Such a system could be used to extract the structure of articulated objects and then perform tasks like recognition, learning and image-based rendering. Even though such a system is still further down the road, this thesis aspires to take one step in that direction.

This thesis only focuses on the first stage of such a system, which is extracting a useful representation from a moving articulated object. Nevertheless, this stage is of great importance. In order to perform any high-level tasks for a given object there must be some understanding of a common denominator: a model that is representative, yet broad enough to encompass different instances of an object.

Throughout this thesis, the term **articulated object** refers to any object for which parts are allowed to move rigidly relatively to others. Each part must be connected to another by a revolute joint that must be visible. Articulated motion differs from elastic motion whose only constraint is the smoothness of motion (see [2] for a more complete taxonomy of motion types).

The method proposed in this thesis borrows from well established theories in knowledge representation to propound a model for articulated objects based solely on their joints. It uses evidence from experiments in psychology to argue that such a representation is appropriate for tasks like learning, tracking and categorization. This thesis introduces a method based on efficient geometric matching that is able of extracting the articulations of objects from a stream of images and tracking them in time. This methodology is inspired from developments in cognitive science and clearly breaks away from knowledge-based tracking where *a priori* knowledge heavily restricts versatility. To make this method as general and adaptable as possible, not a single compromising assumption is made.

The method goes as follow. First the moving edges of the object are found. For this task, a novel method that combines high-order moments, interframe differences and edges from three consecutive frames is proposed. Then, the Hausdorff distance is used to decompose the moving edges into rigid components by performing partial matching on the moving edges. Thirdly, the overlapping edge segments between two adjacent components are found by processing each rigid component. Finally, these overlapping segments are tracked by using a deterministic version of the CONDENSATION algorithm.

The contribution of this thesis is a step towards a system able to perform
high-level analysis of complex motion. Even though the project aims at tackling problems intrinsic to articulated objects, it is hoped that the arguments supporting this thesis will spill on to other areas of computer vision.

The thesis is presented as follows: Section 2 reviews the major efforts made in tracking articulated objects and the human body. Section 3 reviews theories in knowledge representation and proposes a representation suitable for general tracking, categorization and learning of articulated objects. Section 4 outlines the steps of the methodology for tracking the joints of an object. Each of the individual steps of the methodology will be described in sections 5 through 9 and are broken down as follows: Section 5 explains how the moving contours are extracted, then section 6 shows how an articulated object can be decomposed into its limbs by using the Hausdorff distance. The computation of the Hausdorff distance is summarized in section 7. Section 8 describes how the articulations can be simply extracted from the decomposition. It also shows how to create a second set of joint observations for each frame and also describes how a deterministic flavor of the CONDENSATION algorithm is used to integrate both sets of observations while tracking the articulations. Section 9 presents results obtained by the system. Finally, section 10 and 11 discuss of future work and conclusions on the project.
Section 2

Previous Work

The proliferation of publications and systems for tracking articulated objects is a good indicator of the rising popularity of the area. Many surveys were published in the late 90s and the reader is invited to read those of Gavrila [20] and Aggarwal [1, 2] for complementary details.

Systems for tracking articulated bodies can be divided according to different criteria [1]. For example, some methods aim at tracking limbs while others at tracking deformable objects as whole. Since this thesis aims at tracking body parts, this review will reflect previous research in that specific area.

A second criterion, orthogonal to the first one, also divides methods in two categories. Methods in the first one use *a priori* knowledge, which is some sort of information about the viewed object, while the others do not use any. However, the line separating the two categories is somewhat fuzzy and depends on one’s definition of knowledge. Often, in the literature, *a priori* knowledge refers to a predetermined visual representation of an object (a *shape model*) and obviously such methods are categorized as knowledge based. However, some methods might be classified as not knowledge-based despite using primitives for approximation purpose (i.e. lines, blobs and deformable models). Whereas these primitives should be considered as knowledge is debatable. Even though primitives are less restrictive than shape models, one might argue that they constitute a form of knowledge because they restrict the possible shapes. For example, the work described in [37, 38], which this thesis is inspired from, uses deformable models [50] to approximate the contours of limbs (this will be explained in more details in later sections). In that case, the system does not have any knowledge about the shape of the limbs, how-
ever it tries to parameterized the shape with an analytical model. Throughout this thesis, such methods and any other method using constraints or making any assumptions about the object will be considered as knowledge based. The next sections will review methods (with and without knowledge) for tracking articulated objects. Many of these methods were specifically designed to track the human body but for the objectives of this thesis the distinction is not necessary.

2.1 Tracking With \textit{a priori} Knowledge

While researchers have developed methods to recover structure from rigid motion without \textit{a priori} knowledge, most of the methods for articulated motion analysis still require knowledge about the object. The kind of knowledge most often used consists of visual representations (shape models). The conventional procedure such methods rely on consists of the following steps. First, define a shape model of the object; usually, shape models are 3D, 2D, stick figure or a combination of those, but some are more exotic. Then, extract features from the image. Finally, fit the model to the features.

For the objectives of this thesis, all the following knowledge-based methods have the handicap that they are not designed for bottom-up approaches. Having the shape of the object beforehand, and often combined with the initialization of the shape in the image, make these methods ill adapted for the problems this thesis aims to tackle.

2.1.1 3D Shape Models

Perhaps the first attempt to use a shape model for tracking the human body was by Marr and Nishihara [47] in which their model consisted of truncated ellipsoids. Hogg [25] and Rohr [56] have also used models similar to this one. In [25], a combination of the contours and regions is used for image features and the matching algorithm is a least mean square. In [56], the 3D position of a walking human is found by using (walking) dynamic motion models from medical data. Then, the contours of a model composed of 14 truncated ellipsoids are compared to edges of the image. Finally, tracking is performed by using a Kalman filter with the parameterized motion model. In addition of being knowledge-based, this system is furthermore limited to track
a given periodic motion.

In Wachter's research [63], the contours of a model with 12 DOF and 13 cylinders are fitted to the edges in the image. Tracking is performed by assuming a constant velocity motion model for all limbs and using an iterative extended Kalman filter. The robustness of the state updating is increased by including the intensity information of the limbs regions and additional knowledge in the form of anatomical constraints (interval of valid values for each joint and acceleration).

Rehg and Kanade [55] use cylinders to represent two fingers of a hand. They use an invariant visibility order to model occlusion between the fingers. Templates of the finger parts (i.e. 2D projection of a fingertip) are tracked by gradient descent.

Also in the volumetric category, Deutscher et al [14] use 17 truncated conics to represent an elaborate model of a human with 29 DOF. The system starts by extracting edges and the foreground silhouette resulting from background subtraction. For both types of features, a sum-squared difference is used to determine how well a state of the model fits the image. Then, the system tracks the object with annealed particle filtering.

### 2.1.2 2D Models

With 2D or flat models, each limb is represented by a 2D area. These methods suffer from the fact that the models can only represent one view of the object. To get around this problem some systems use a set of models, each representing a different viewing angle of the object. Though there has not been any report of a system using successfully more than one flat model during the same tracking sequence (automatically selecting the appropriate model depending on the view). Which means that all of the following methods only track objects under a constant viewing angle.

In a recent paper, Felzenszwalb and Huttenlocher [17] perform pictorial matching by using the colors of a cardboard (rectangle limbs) model. Their paper addresses issues that are more general than articulated objects but they have successfully used their method to estimate the pose of the human body. The fitting algorithm finds a global optimal match by minimizing two cost functions: one that measures the color intensity similarity and one that measures how well the relative positions of the limbs are consistent with the model.
(i.e. two adjacent limbs far apart would give a high cost). The optimal configuration is found by using dynamic programming by first fitting a limb, then finding the best fit for its adjacent limbs and so on.

Ju et al [36] developed a cardboard model composed of connected quadrilaterals. They represent an articulation by four vertices of two adjacent quadrilaterals (two vertices by limb). Each limb is connected to its neighbor by a pair of "ligaments" that connect vertices of adjacent limbs. The optimization is carried by gradient descent that uses robust statistics to minimize the cost function of both the stretching of the ligaments and the optical flow in the quadrilaterals.

Leung and Yang [40] use a silhouette composed of 2D ribbons as their model. First, the moving contours of the object are extracted by interframe difference combined with detected edges. Then, by analyzing the shape and motion, the method determines which ribbons belong to the object and the parts of the body are labeled according to the model.

### 2.1.3 Stick Figures

Chen and Lee [10] use a model with 14 joints and 17 limbs. The fitting starts by finding the head using six known features. Once the head is located, constraints (smooth motion and physical constraints provided by the model) are used to search for other parts of the body and to prune impossible poses.

Akita [3] combines a stick figure with a volumetric model composed of truncated conics. A set of stick figures, called *key frames*, is used to represent typical poses of the human body during a predetermined activity (walking). Two methods are used to track the body. The first one assumes small motion in which case the body parts are tracked by using the volumetric model and the contours of the object. If this fails, the key frames are used to re-find the body in the image.

### 2.1.4 Others

Niyogi [53] developed a clever system for gait analysis that also allows the recovery of the contour of the object and the extraction of its stick figure. Niyogi's system analyses separately the intensity changes of each line of the image over time (the XT plane) and then fits a snake to the braided traces of
limbs in that plane. Even though the system does not use a visual representation of the object it does use a model of the motion for detecting the presence of the object. This motion template is also used to initialize the contours of the object, which is then used to build the stick figure. Also, the method assumes knowledge about the locations of known joints (i.e. knees, hips and ankles for a human) as well as knowledge to help locate the head and the feet. Finally, the method is not incremental since it needs to process a whole sequence as input.

2.2 Tracking Without *a priori* Knowledge

Unfortunately, very few attempts have been made to track articulated motion without using knowledge. Certainly, the work of Huttenlocher *et al* [30] falls in this category. This method, along with the research of Kakadiaris *et al* [37] (which uses deformable models) will be detailed later.

Black and Jepson [6] extract the Eigen vectors of a set of representative training images by using SVD. When an input image is projected on these Eigen bases, the system can determine which training image is matched. This method is extended to perform matching under a parametric transformation. Tracking is performed by the parameterized matching which recovers the motion between frames. This method has been used successfully to track and recognize simple hand gestures. The method can only track motions for which it was trained for and also depends on the irradiance of the object.

Webb and Aggarwal [33] found a method for recovering the structure of an articulated object from stimuli similar to moving light displays. They assume that the traces made by points attached to the limbs are available and that the articulated motion is around a fixed axis. By using the fixed axis assumption they prove that the trace of a point attached to a limb is projected to an ellipse on the image plane. From these ellipses, the structure of the object is recovered up to a reflection.

Isard and Blake [32] presented a stochastic algorithm called CONDENSATION based on particle filtering. They show how a B-Spline corresponding to the contour of a deformable object can be tracked by using a Markovian framework. Their method does not require any knowledge about the shape. Though, the B-Spline must be initialized on the contour of the object beforehand and the use of a Markov motion model requires some knowledge about
the motion of the object.
Section 3

Identification vs. Categorization

As mentioned in the introduction, the motivation for this thesis is more about accomplishing high-level tasks with articulated objects than it is with tracking them. It is an effort to use tracking information to extract the structure of complex moving objects. Therefore, tracking is the means not the goal.

To grasp the intentions behind this idea, one must understand the distinction between two tasks that require different representations: Identification and categorization\(^1\) [16]. The next sections will establish the distinctions between the two forms of recognition. The goal will then be to extract such a representation from streams of images.

3.1 Identification

The process of identification determines which particular object is being viewed. It identifies a car from all other cars, a particular face from all other faces, a certain shoe from all other shoes.

Identification systems often use features like corners, regions, scale invariant features, edges or any other visual cues available. Methods to achieve identification typically consist of the following steps: Modeling an object based on such features, then extracting features from the image and finally, finding a match between the two sets of features.

These features reflect the appearance of the object and using models composed of such features means that any object that is visually different

\(^1\)Categorization is also often known as generic object recognition. However, for the remainder of this dissertation the term categorization, as defined in [16], will be used.
should not be match.

3.2 Categorization

Unlike identification, categorization is not about identifying a specific instance among others. Instead, the aim is to classify the objects. It is important to note that if for identification the system must have seen an object before identifying it, this is not required for categorization. For example, a system should be able to recognize a particular human body without requiring seeing it beforehand.

Just like for identification, an appropriate representation is required to categorize objects. However, whereas identification uses a representation based on image features, such representation is not appropriate for categorization. Otherwise this would imply that all instances of a class look alike. Instead, a form of abstraction is required, that is a category should represent the common denominator of all possible instances of that category.

Many theories of representations have been proposed for the purpose of categorization in computer vision [16]: Structural decomposition, geometric constraints, multidimensional feature spaces, isomorphism, mental models and of course shape models like the ones mentioned in section 2.

3.2.1 Structural Decomposition

Structural decomposition represents an object by approximating its main aspects and their spatial relationships. An example of structural decomposition is the recognition by components introduced by Biederman [5] that describes objects with generalized cylinders (called geons). However, for deformable and articulated objects, the spatial relationships between components can change in which case RBC fails to account for such objects. Also, the representation highly depends on the visual aspect of an object and some objects seem to be too problematic to be represented with geons. Finally, Pylyshyn [54] argues that RBC is a knowledge-based method since it assumes prior knowledge of possible components. Nevertheless, as it will be explained shortly, an appropriate representation of the structure of an object can be adequate for articulated objects as long as it does not represent visual aspects of the object (like geons).
3.2.2 Geometric Constraints

Geometric constraints representation is based on “pinning down” features of the model to the corresponding ones of the image. When a feature is pinned down, it creates a constraint on the possible locations of the other features and hence on the pose of the object [44]. Representation by geometric constraints is broadly used for identification tasks since it can be successfully combined with image-based features. Unfortunately, if the geometry of the object is allowed to change, as it can with articulated objects, this representation is not as effective. For example, even though pinning down a feature belonging to a left arm does constrain the possible locations of other features, the search space remains very large. Thus, more features need to be pinned down and the search space for those features is large.

3.2.3 Multidimensional Feature Spaces

Multidimensional feature spaces are the middle ground between recognition by components and geometric constraints. Instead of making coarse approximations like structure decomposition and relying on precise features like geometric constraints, multidimensional feature spaces integrates these two modalities to perform recognition. For example, an object can have the same shape while having different textures. This representation combines the strength of both representations. However, it fails to cover their weakness relative to deformable objects.

3.2.4 Second Order Isomorphism

In [16], Edelman argues that a categorization system should represent similarities between shapes (second order isomorphism) instead of the geometry of objects (first order isomorphism). Basically, a first order isomorphism representation encodes a one-to-one relation between an object shape and its representation. A second order isomorphism representation encodes the difference between two shapes instead of the shapes themselves. To represent the changes in shapes, Edelman uses a metric over a shape space. The parameters of the shape space are defined by fiducial points that determine the shape of the object. Despite the soundness of Edelman’s theory for rigid objects his definition of shape space does not account for deformable objects. Accord-
ing to his theory, taking two different poses of the same articulated object would yield a high dissimilarity. For Edelman’s theory to work for articulated objects, a parameterization invariant to deformations should be defined (or analogously a metric invariant to such deformations). Even though this thesis does use a metric (Hausdorff distance), it is only used as a tracking tool to decompose an object.

### 3.2.5 Shape Models

Shape Models like those mentioned in section 2 are actually worst for categorization than those just presented. Whereas the previous representations can be derived from raw images, the shape models of section 2 are approximations of real objects. One drawback of this is that using distortions of real objects can only downgrade the accuracy of the matching process. Furthermore, it is in general not true that one shape model fits all the instances of a category. Perhaps in practice all humans look alike. However, biological vision capabilities are not only limited to “real world” situations. Anybody can watch cartoons or animations in which characters have visual aspects different from their “real world” appearances but it is still possible to categorize the characters.

### 3.2.6 Mental Models

Perhaps, one of the most influential recent theories in psychology and cognitive sciences is the mental models [35]. A mental model corresponds to the simplest structure of what they represent. They are often described as mental images of concepts, but this is not a requirement since many phenomena cannot be visualized [35]. Mental models define the structure corresponding to the concept they are representing. Since they are internal representations it is not clear what constitute a good model or how to define one. However, the following guidelines describe the idea behind mental models:

1. Structural (how it works, describes the internal mechanics of a device in terms of its component parts)
2. Functional (how to use it, procedural knowledge about how to use the device)
For representing an articulated object, structural mental model seems the obvious choice.

Other evidences to support the structural representation can be found from various sources. A first one is how children perceive the human body [60, 51]. Everywhere around the world children draw humans the same way: a stick figure with a head on it. For children this is what a human is all about: a kinematical structure with a ball on top of it. Since the information of a stick figure can be encapsulated by the configuration of the articulations this is all the information needed

Perhaps an even more convincing argument in favor of articulations as a representation comes from the seminal research on moving light displays (MLD) by Johansson [34]. In his work, he found that humans are able of recovering the structure of an object just by tracking a few points on its surface. Moreover, his results also show that humans are able of recognizing human motion only by the movement of articulations and endpoints

Since this thesis focuses on articulated objects in general, there is only a small step to generalize and consider a jointed representation as an appropriate representation for any articulated object. From a vision point of view the question is how to extract such a representation from images.

3.3 Bottom Up vs. Top Down

As mentioned in section 2, many great efforts were made for tracking articulated objects by using prior knowledge. It was explained that the weakness of such systems is their lack of generality and adaptability because they need to know something about the object before analyzing it. In psychology and cognitive science literature this methodology, in which perception is initiated from knowledge, is called top-down information flow [11]. Using a top-down process is problematic for general tasks like tracking arbitrary objects. Given a knowledge base, how do you know which model to use for tracking? This would either imply searching a potential match for all the models in the knowledge base or designing a bottom-up information flow method to extract the

2As for the role of the head when determining if the object is a human or not, one could argue that the head gives the "humanity" to the body. That is, a human body without a head would still look very much like a human from a visual point of view. However, nobody would ever agree that a human walking around without a head is a human.
model from the images.

Philosophers have been arguing for centuries (millenniums) on how human perception works. Nowadays, this discordance remains pretty much the same and only the vocabulary has changed. Scientists now talk about *cognitive penetrability* [11] versus *cognitive impenetrability* [18] of perception. For the problems in vision this would amount to say: Do we see things only because our brain processes the retinal signal or because we expect to see things in certain situations?

Despite the fact that many evidences support cognitive penetrability in visual perception (like reentrant pathways, see [54] for a short review), some scientists still believe that there exists a uniquely visual system similar to Marr's definition of early vision [46] and that this system is cognitively impenetrable. Indeed, many indications resulting from experiments point in the direction of impenetrability of perception [54]:

- Some experiments show that knowledge do not influence perception as with the case of perceptual illusions: no matter how often you look at the Müller-Lyer illusion, one bar always seems longer than the other, even if you know they are of the same length (same thing can be said about other illusions).

- Experiments have shown that perception can be resistant to expectations and physical properties.

- Advances in neuroscience conclude to independence of parts in the vision system.

Following this, it can be said that believing that knowledge-based tracking systems like the ones mentioned in section 2 can be used for arbitrary object is a pious wish. The reasons are the following: They do not use appropriate representations or information processing flow. Therefore, to avoid these issues the system must use the most elementary representations and it must be able to extract them from images.
Section 4

Methodology

The method for locating joints presented in this thesis is greatly inspired from the works of Kakadiaris [37] and Huttenlocher [28]. Kakadiaris locates articulations by first fitting a deformable model to the contour of the viewed object. Initially, a geometric primitive (super-ellipsoid) is fitted to the whole contour of the object by applying a global deformation to the model followed by a local deformation to fit the contour more or less exactly (figure 4.1 a). The global deformation allows the model to deform and bend according to the overall shape of the object. The local deformation handles the smaller variations on the contour of the object. As the object bends, the model tries to adapt itself to the bending contour by deforming itself accordingly. However, beyond a given "bending" threshold, the model is decomposed in three subparts: a fixed zone, a bending zone and a relocation zone (figure 4.2). Hence, the bending zone accounts for the bending deformation. By monitoring the bending parameters of the model and comparing them to a threshold, the system can decide when a piecewise decomposition is required. If it is, two new deformable models are initialized on the resulting fixed and relocation subparts (figure 4.1 b) and the joint is found at the intersection of these two models (figure 4.1 c).

The piecewise decomposition of an articulated object is at the base of the method developed in this thesis. However, a different approach on how to perform the decomposition is proposed. Even though the use of deformable models is an improvement over shape models for general tracking, it is still a model, which implies limitations on the shape of the object.

In order to remove the limitations of deformable models any form of knowledge or assumptions must be avoided. Not only does this includes the
Figure 4.1: Intersection of two regions enclosed by deformable models.

Figure 4.2: Piecewise decomposition of an articulated object.
complex shape models of section 2 and the deformable models of Kakadiaris, but also any kind of parameterized primitives like line segments, curves, blobs and surfaces. Instead, let the world be its best representation and use information from the image as it is, as in the work presented by Huttenlocher et al [28]. The method presented by Huttenlocher searches the transformation space to minimize the Hausdorff distance between a model of the object and an image. Beyond the nice properties of the tracking algorithm, the interesting strength of this method is that the model is derived from the intensity image such that neither a priori knowledge nor parameterization is necessary. The model is composed of the edges of the object and the tracking algorithm looks for a good correspondence between the model at time $t - 1$ and the edges of the whole image at time $t$. Despite the fact that Huttenlocher has used this algorithm to track non-rigid objects, the system is suited to track rigid objects and it tolerates only a small amount of non-rigid deformation.

The box diagram illustrated in figure 4.3 outlines the proposed system. The following sections will describe in detail the method at every step and the tools used to perform the required tasks.

First, the contour of the object, composed of moving edges, is extracted at each frame. Then, geometric matching is used on two consecutive contours to decompose the object in rigid components (i.e. arm, leg, torso) as shown in figure 4.4. The idea is to align each rigid component individually. First, a rigid component (typically the largest one) is matched by using the Hausdorff distance a). Then, the matched features are removed b) and the matching algorithm is performed again on the remaining features in order to match a second rigid component c). This procedure is repeated until no more matching is possible. At the end, the joints are indicated by the remaining edge segments of the decomposition procedure d). Actually, this method is slightly modified and will be improved with special processing on adjacent components to avoid false positives (as the one shown in figure 4.4). Finally, the observations resulting from joint detection are tracked in time by merging sets of observations...
Figure 4.4: Joint detection.
and using a deterministic version of the CONDENSATION algorithm.
Section 5

Feature Extraction

The first step consists of extracting the features of the moving object. Over the years, researchers in computer vision have developed a multitude of different features for representing scenes. To locate and track articulations some features are more appropriate than others and trade-offs between accuracy and manageability are expected. The next sections will argue which features are suitable for the tracking task and how they can be efficiently extracted.

5.1 Regions vs. Edges

Regions provide perhaps the best information for localizing articulations. In his work, Kakadiaris [37] shows how an articulation can be located by finding the intersection of two regions, each corresponding to a limb enclosed by a deformable model (figure 4.1). However, as argued earlier, deformable models imply constraints on the possible shapes of the object and hence, unsuitable for this thesis. Therefore, the usage of regions depends on region segmentation from motion, which to this day is still a major challenge in computer vision.

Edges are much easier to find, but the information they provide is rather poor. To locate joints by using edges, the edges belonging to the jointed regions must be detected as in figure 4.4. Unlike regions, finding the overlapping edges over an articulated area is not a trivial task because of the small area covered by edges. Also, edges can only detect the boundaries of an articulation, which means that the observation of an articulation may be fragmented. Nevertheless, because of the easiness to extract them, edges are the chosen features for this thesis. It will be shown later how the disadvantages of edges...
5.2 Contour Extraction

The method for extracting contours used in this thesis is based on two concepts: interframe difference and high-order statistics. The method combines approaches used to filter moving objects from the background [52] and extract moving edges [39].

5.3 Interframe Difference

Interframe difference is probably the simplest method to segment a moving object. It is simply defined by \( D(t) = I(t) - I(t - 1) \) where \( I(t) \) represents the image at time \( t \). However, the regions segmented by this method are ambiguous. For example, if a disk is moving to the left as shown in figure 5.1, the result of the interframe difference \( I(t - 1) - I(t) \) will be two disks and background noise. Simple interframe difference does not tell which disk represents the position of the object at time \( t \). To solve this ambiguity problem three consecutive images are used \( I(t - 1), I(t), \) and \( I(t + 1) \). The interframe differences \( I(t - 1) - I(t) \) and \( I(t) - I(t + 1) \) are computed and then combined with a logical \( \text{AND} \). The result shows the correct location of the moving object at time \( t \). One drawback is that this method does not get rid of the
background noise and the segmentation depends on an arbitrary threshold. Another drawback is that this method depends on the overall illumination of the scene. Finally, the object must move in three consecutive frames otherwise it will go undetected. The fact the system is a non-causal can also be considered as a disadvantage.

To improve segmentation, the system uses a well-known method to filter the background noise. It is well known that a Gaussian can be fully parameterized by its mean and variance and that all the moments above the variance are equal to zero. By modeling the background noise resulting from the inter-frame difference as a Gaussian it is possible to extract the structured signal at very low expense.

There are many methods available to filter Gaussian noise [19] depending on how much is known about the structure of the signal to be recovered. The most general one is for when the structured signal is totally unknown. Then, the method consists of using high-order moments to discriminate noise from structure. By computing the $n$th moment, where $n > 2$, of the signal over an area, the resulting value will be zero (or close to zero) if Gaussian noise is being detected, otherwise a large value will indicate that a structured
Figure 5.3: Detection of moving edges.

signal is detected. This simple method can be used in images by computing the nth moment over a small 2D patch [52]. Figure 5.2 illustrates an example of using this method. Notice that the disks become circles since the inside of a disk after the interframe difference is a Gaussian signal with a different mean.

There is a trade off that must be considered for the size of the patch. This trade off is analogous to the trade off for convolution operators: Using a large patch decreases localization while a small patch is less representative than a large one. The poor localization is represented in figure 5.2 by the thick contours of the circles.

5.4 Moving Edges

The interframe difference of figure 5.1 detects motion, has a good localization, but is corrupted by noise. Filtering with high order moment also detects motion but decreases localization. Hence, combining both results with a logical AND retains the best of both methods. Furthermore, by taking this result and applying a logical AND with the edges detected by an (Canny) edge detector
[9] at time $t$ results in moving edges, as shown in figure 5.3. The use of edges allows the contours to represent exactly the shape of the object and results in narrower contours (which is beneficial to the matching algorithm described in the next section).
Section 6

The Hausdorff Distance

The Hausdorff distance was introduced to the computer vision community in a series of articles by Huttenlocher in the early 90’s [26, 28, 31, 27, 30, 29].

Whereas the Euclidian distance is a metric that measures the distance between two points in Euclidian space, the Hausdorff distance is a metric that measures the distance between two sets of points. Hence, the Hausdorff distance has the ability to determine how much two sets are “close” to each other or, in other words, how similar they are. Given two sets $A$ and $B$, a metric $d$ must have the following properties

1. $0 \leq d(A, B) \leq \infty$
2. $d(A, B) = 0$ if and only if $A = B$
3. $d(A, B) = d(B, A)$
4. $d(A, B) \leq d(A, C) + d(C, B)$ (triangle inequality)

and it was shown that the Hausdorff distance satisfies all four [26].

Let $M$ and $I$ be two sets of points corresponding to features of a model and an image respectively. In this thesis, the features are the moving contours found with the method presented in the previous section. The Hausdorff distance is defined by

$$H(M, I) = \max(h(M, I), h(I, M))$$  \hspace{1cm} (6.1)
Forward Hausdorff distance

Reverse Hausdorff distance

\[ h(M, I) = \max_{m \in M} \min_{i \in I} ||m - i|| \]
\[ h(I, M) = \max_{i \in I} \min_{m \in M} ||i - m|| \]

are called directed Hausdorff distances and \( ||\cdot|| \) is the \( L_2 \) norm. More precisely, \( h(M, I) \) is called the forward Hausdorff distance and \( h(I, M) \) is called the reverse Hausdorff distance.

The intuition behind the directed Hausdorff distances is depicted in figure 6.1. Let's begin with the forward Hausdorff distance \( h(M, I) \). For each point \( m \in M \), let \( f(m) \) be the distance between \( m \) and its closest point \( i \in I \). Then the forward Hausdorff distance is equal to the greatest value of \( f(m) \). In other words, if \( h(M, I) = d \), then every point in \( M \) is within a distance \( d \) of a point in \( I \). The reverse Hausdorff distance simply reverses the role of the sets.

It is easy to see that a small value \( h(M, I) \) does not imply a small value \( h(I, M) \). While every point in \( M \) may be close to a point in \( I \) (possibly the
same one) a point in $I$ may be far from any point in $M$. The fact that many points in $M$ may be closer to the same point in $I$ explains why the Hausdorff distance does not establish a one-to-one correspondence. This is illustrated in figure 6.1 where the points $I_1$ and $I_2$ both have the point $M_1$ as their closest neighbor.

The Hausdorff distance $H(M, I)$ is simply the greatest value between $h(M, I)$ and $h(I, M)$. If $H(M, I)$ is small then every point in $M$ is close to a point in $I$ and vice-versa. However, if only one point in $M$ is far from any point in $I$, or vice-versa, $H(M, I)$ will be large. Finally, if $H(M, I) = d$, then every point in $M$ is within a distance $d$ of any point in $I$ and vice-versa.

The Hausdorff distance has many properties that make it a very attractive tool for the problem decomposing a moving articulated object. Here is a short summary of the most relevant ones to this thesis.

The Hausdorff tracking method presented in [28] creates the set $M$ directly from detected edges in the image, such that elaborate predetermined man-made shape models can be avoided. This is the main reason why the Hausdorff distance is an appropriate tool for bottom-up approaches.

Hypothesis and test methods make sure that the features of the model
match a region of the image (hypothesis) but also that the features within that region match the model (test). The Hausdorff distance can be viewed as a hypothesis and test method for which the forward distance expresses the hypothesis part and the reverse distance expresses the test part. Hypothesis-and-test methods have a clear advantage over methods that only match a model in an image. For methods that only use the hypothesis part, if a region of the image contains clutter many models could be mismatched in that region. However, if every feature of that region is also near a feature of the model, then, unless the model has dense clutter too, the mismatch will be rejected. Figure 6.2 illustrates this. The model could well match the two shaded areas since it can be translated such that every point of the model is close to a point of the image. However, the area A matches the model better than B because many points in B (those from the clutter in the middle) are far from matching any point of the model. In this case, any method that only performs the hypothesis part is prone to generate a false-positive.

Invariance under a group of transformation is another important characteristic of a metric. For a metric $d$ and a transformation group $G$, a metric is invariant under $G$ if $d(g(M), I) = d(M, I) \ \forall g \in G$. In other words, invariance allows a metric to measure the same distance (similarity) even if the model has been deformed by any transformation in $G$. Theoretically speaking the Hausdorff distance is only invariant for isometries ($Iso$). However, by constraining the deformation parameters, such that degenerate cases are avoided, the Hausdorff distance has been successfully used under the affine group [23, 58].

Perhaps the most attractive characteristic of the Hausdorff distance for this thesis is its tolerance to noise and outliers. In its original form, the Hausdorff distance actually performs very poorly in the presence of outliers. In fact, it measures the greatest outlier. However, a modified version, called the partial Hausdorff distance, can discard a portion of the features in the model and the image as outliers. The partial Hausdorff distance only considers the best matching portion between features of the model and the image. The fractions of the number of features to be considered are (unfortunately) given as input parameters. It is important to note that these parameters can have any value. Unlike robust statistics where the maximum proportion of outliers that a method (least median of squares) can handle is 50% [57], the partial Hausdorff distance can accommodate a given proportion of outliers above 50%. 29
Outliers can be the result of many circumstances like occlusion and noise. Most interestingly, outliers can be the consequence of non-rigid motion of parts of an object. In Meier's work [48, 49], the partial Hausdorff distance is used to match the largest part of a walking human (torso) while the outliers are the features belonging to the limbs that move relatively to the torso (arms and legs). This behavior of the partial Hausdorff matching is exactly what must be used for the piecewise decomposition of an articulated object.

The Hausdorff distance has also many other nice properties that are somewhat less relevant for this work and the reader is referred to [58, 22, 62] for more details.

6.1 The Partial Hausdorff Distance

As mentioned above, the partial Hausdorff distance addresses the problem of outliers by using ordered selection instead of the max operator. For $0 \leq K \leq |M|$ and $0 \leq L \leq |I|$ the partial Hausdorff distance is defined by

$$H(M_K, I_L) = \max(h_K(M, I), h_L(I, M))$$

and

$$h_K(M, I) = K^{th} \min_{m \in M, i \in I} ||m - i||$$

$$h_L(I, M) = L^{th} \min_{i \in I, m \in M} ||i - m||$$

where $K^{th}$ and $L^{th}$ are rank values. Usually, $K$ and $L$ are expressed in terms of fractions of the number of features in the model and the image respectively by $K = f_K|M|$ and $L = f_R|I|$, where $f_F$ and $f_R$ are the percentage of features to be considered as inliers. For example, for the forward distance, if there are reasons to believe that only 70% of a model containing 200 features are visible in the image, then the mismatch distance of the 140th best matched point of the model gives the partial forward distance. The portions of features that are occluded or perturbed are not taken into consideration in the computation of the partial Hausdorff distance. Which features are considered as inliers does not matter for now, the partial Hausdorff distance only tells how good the match is.

One drawback of using the partial Hausdorff distance is that it does not respect the metric properties anymore. This is easily illustrated for the
triangle inequality (metric property 4) by figure 6.3. In the figure, a portion of $A$ matches a portion of $B$ while a portion of $B$ matches a portion of $C$. The triangle inequality says that in this case a portion of $A$ must match a portion of $C$, which is not the case. However, by using weaker properties, the Hausdorff distance has "practical metric behavior" and as such can be used for matching [28].

6.2 Finding the Pose Using the Hausdorff Distance

As explained, the Hausdorff distance measures the similarity between two sets of points in the same coordinate space. For the purpose of matching and pose estimation, the model needs to be transformed to a region of the image that offers the best match. Therefore, equation 6.1 is expressed as a function of a transformation $g \in G$

$$D_G(g) = H(g(M), I)$$

where $g(M)$ is the transformation of the model $M$ under $g$ and $H(g(M), I)$ compute the Hausdorff distance between the transformed model and the image. The task is to find the transformation $g$ that minimizes the Hausdorff distance. This can be expressed as

$$H_G(M, I) = \inf H(g(M), I) = \inf D_G(g)$$
and the directed Hausdorff distance are defined by

\[ h_G(M, I) = \inf_{g \in G} h(g(M), I) = \inf_{g \in G} d_G(g) \]

\[ h_G(I, M) = \inf_{g \in G} h(I, g(M)) = \inf_{g \in G} d'_G(g). \]

An algorithm for how to minimize the Hausdorff distance for the affine transformation group will be explained in section 7.

### 6.3 Box-Reverse Distance

Computing the forward distance is fairly straightforward. All that needs to be done is apply a transformation to each point of the model and look for the closest image point. However, in order to have a valuable reverse distance, only a region of the image must be considered. Remember that the reverse distance is computed from the distance between each image point and its closest transformed neighbor point. If the transformed model only covers a small region of the image, there might be image points outside this region for which the distance to their closest transformed model point is quite large.
and irrelevant. This is illustrated in figure 6.4 where the task is to find the best match of the model $M$ a) in the image $I$ b). Figure 6.4c) shows a good pose of the model. However, because of the points $I_1$, $I_2$, $I_3$ and $I_4$ are far from any point of $g(M)$, the reverse distance, and hence the Hausdorff distance, is large causing $g$ to be a poor candidate. Since $I_1$, $I_2$, $I_3$ and $I_4$ are of no interest, only points within the region of the transformed model are considered. To achieve this, the model is simply enclosed within a box $R$ and only the points within the transformed box $g(R)$ are considered. Therefore, only local characteristics of the image are used in the reverse distance that is now defined by the box-reverse distance

$$h_{box}(I, M) = \max_{i \in g(R)} \min_{m \in M} ||i - m||.$$  

Using a box has the drawback that it can be a poor region of support for the model. However, it makes the already cumbersome computation of the Hausdorff distance more efficient.

To wrap up this section, let's recall that the Hausdorff distance is a metric that offers an intuitive "look like" characteristic. The model is extracted from the image and no parameterization of the features is necessary. The partial Hausdorff distance and the partial box-reverse distance make the distance robust to outliers, occlusion, clutter, noise and other deformations such as non-rigid motion.
Section 7

Computation of the Hausdorff Distance

The motion model used to compute the minimum Hausdorff distance under transformation (section 6.2) is application specific. Algorithms have been tailored for different transformation groups: translation [27, 4], rigid motion [21, 4] and affine motion [31, 23]. The translation model is far too restrictive but a reasonable computation time can be achieved. With articulated objects, the rigid motion model is a minimum requirement. However, affine motion is more general and more appropriate, especially with ball-and-socket joints. Unfortunately, computing the Hausdorff distance under affine motion is very expensive; the lower bound for the exact computation is $\Omega(n^9)$ [58], where $n$ is the cardinality of each set. In practice, a multitude of hacks can be used to speed up the process such that acceptable run time\textsuperscript{1} can be achieved [58]. The method used in this thesis is the multi-resolution method presented in [31]. For self-containment, the next sections will review this method with the terminology adapted for an affine motion model.

7.1 Computing the Hausdorff Distance by Using the Voronoi Surface

The computation of the Hausdorff distance can be alleviated by using a derivative of the Voronoi diagram [13]. In 2D, a Voronoi diagram is a subdivision of

\textsuperscript{1}However, the computation is still very far from real time.
the plane into cells where one site is being assigned to each cell and each point of the plane is assigned to its closest site. The subdivision is made such that every point assigned to the same site belongs to the same cell, as illustrated by figure 7.1.

The Voronoi surface [26] is related to the Voronoi diagram in the following way: The value at each point in a cell is set to the distance between the point and its site and is expressed as

$$\delta_i[x, y] = \min_{i \in J} ||(x, y) - i||$$

where J is the set of sites. In 3D, a Voronoi surface has a shape that looks like an egg carton. The Voronoi surface is often called a distance transform and
can be computed in different ways. One method uses the chamfer distance [7] that approximates the $L_2$ distance up to a precision of 0.06 on an area of $(M+1)(M+1)$. An example of how to compute the chamfer distance is shown in figure 7.2. The masks a) and b) are successively translated over an image in which sites have the value zero and other pixels are set to infinity. The mask a) is moved from left to right and top down and the mask b) is moved from right to left and bottom up. At each pixel the value $d_1$ and $d_2$ are added to their underlying pixel value and the new distance of the zero pixel is the minimum of the five sums. Using the integers $d_1 = 3$ and $d_2 = 4$ instead of optimal real value slightly deteriorates the accuracy of the distance to 0.08M.

Once the Voronoi surface is computed, computing the Hausdorff distance becomes straightforward. To compute $h(M, I)$, first compute the distance transform of $I$, $\delta_I$, and then probe $\delta_I$ at each point $m \in M$. To visualize this, imagine that $M$ is superposed on top of a Voronoi surface. At each point $m \in M$ we look straight down at the point on the Voronoi surface and take this value as the distance from $m$ to the closest point $i$ in $I$ (which point $i$ does not matter). Selecting the greatest of these values gives $h(M, I)$. Similarly, $h(I, M)$ is computed by using the distance transform of $M$ instead. For the partial directed Hausdorff distance $h_K(M, I)$, the $K^{th}$ ordered value is selected instead of the maximum value. An algorithm for general selection in linear time can be found in [12].

7.2 Computing the Hausdorff Distance Under Transformation

Computing the Hausdorff distance under transformation happens to be much more complicated than simply computing the Hausdorff distance. Several algorithms were proposed, but many of them are specific to motion models that are too restrictive for the problems addressed in this thesis. The best algorithm for the general case of affine motion is a branch-and-bound algorithm presented in [31, 58]. Hagedoorn and Veltkamp also developed a branch-and-bound algorithm but with a different lower bound [23]. However, there is no mention how their algorithm can be extended to the partial Hausdorff distance.

A branch and bound algorithm can be defined as follow. Suppose a function $f(x)$ needs to be minimized over a given domain. To apply a branch
and bound algorithm, one must be able to compute the lower and the upper bound of \( f(x) \) over the given domain. The algorithm starts by computing these bounds on the initial domain and, if they are equal, the solution is given by these bounds. If not, the domain is split into disjoint intervals, each a partition of the original domain, and both bounds are computed for each interval. The procedure is applied recursively on each interval, which creates a search tree, until a solution is found.

To speed up the computation, it is advantageous to prune partitions whenever possible. For example, if there is a criterion \( C \) on the lower bound that specifies the upper bound of an acceptable solution, that is \( \min f(x) \leq C \), any interval for which the lower bound is greater than \( C \) can be rejected since no solution in that interval will respect the criterion.

Such an algorithm is used to compute the Hausdorff distance under affine transformation. The algorithm starts by initializing a domain for each parameter. Then, a criterion \( C \) must be specified such that its value represents the minimum for which the Hausdorff distance corresponds to a match (i.e. any distance above this criterion is considered as an unacceptable match). Last but not least, a lower bound is needed. The lower bound used in this thesis is the one used in [31]. A short summary of the lower-bound is given for self-containment.

The algorithm is based on the **transformation uniformity condition**. Given an affine transformation, the parameters of the transformation can be divided in those affecting the \( x \) coordinate and those affecting the \( y \) coordinate. For an affine transformation

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  g_{x1} & g_{x2} \\
  g_{y1} & g_{y2}
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix} + \begin{bmatrix}
  g_{x3} \\
  g_{y3}
\end{bmatrix}
\]

the parameters \( g_{x1}, g_{x2}, g_{x3} \) affect the \( x \) coordinate and \( g_{y1}, g_{y2}, g_{y3} \) affect the \( y \) coordinate. The transformation uniformity condition stipulates that for two neighbor transformations \( g1 \) and \( g2 \) in a rasterised transformation space, the difference in the mapping of a point \( m \) is at most one unit in the \( x \) or \( y \) coordinate

\[
g1(x, y) - g2(x, y) \leq (1, 1).
\]

This implies that for a given domain \( M \), the transformation space must be rasterised such that the transformation uniformity condition is respected for every element of that domain. As a rule of thumb, the greater are the
coordinates of the elements of the domain, the finer is the rasterisation of the transformation space.

Let \( R \subseteq \text{Aff} \) be a rectangular subspace (a cell) of the affine transformation space

\[
R = [l_{g_{x1}}, h_{g_{x1}}] \times [l_{g_{x2}}, h_{g_{x2}}] \times [l_{g_{x3}}, h_{g_{x3}}] \times [l_{g_{y1}}, h_{g_{y1}}] \times [l_{g_{y2}}, h_{g_{y2}}] \times [l_{g_{y3}}, h_{g_{y3}}]
\]

where each interval represents the range of values that a given parameter can take (\( l \) being the lowest and \( h \) the highest). Also, let the center of a cell be

\[
g_c = \left[ \frac{l_{g_{x1}} + h_{g_{x1}}}{2}, \frac{l_{g_{x2}} + h_{g_{x2}}}{2}, \frac{h_{g_{y1}} + l_{g_{y1}}}{2}, \frac{l_{g_{y2}} + h_{g_{y2}}}{2}, \frac{h_{g_{y3}} + l_{g_{y3}}}{2} \right].
\]

In [31] it is proved that a cell can be rejected if

\[
h(g_c(M), I) > C + \left( \left\lceil \frac{h_{g_{x1}} - l_{g_{x1}}}{2} \right\rceil + \left\lceil \frac{h_{g_{x2}} - l_{g_{x2}}}{2} \right\rceil + \left\lceil \frac{h_{g_{y1}} - l_{g_{y1}}}{2} \right\rceil + \left\lceil \frac{h_{g_{y2}} - l_{g_{y2}}}{2} \right\rceil + \left\lceil \frac{h_{g_{y3}} - l_{g_{y3}}}{2} \right\rceil \right)
\]

where the second term in the right hand side expression represents the radius of a cell. The radius of a cell determines an upper-bound of the difference in the \( x \) and \( y \) direction between the mapping of the transformation at the center of the cell \( g_c \) and the mapping of the farthest transformation \( g \) from \( g_c \) such that \( g \in R \) (typically \( g \) is a corner of \( R \)). That is, by using the transformation uniformity condition to bound the difference between the mappings of two neighbor transformations, the difference in the mappings of the transformations in a cell can be bounded as well. In other words, this formulation relates the parameters in the transformation space to their mapping in the plane.

Suppose that the radius of a cell is \( r \). The intuition behind this is that if \( H(g_c(M), I) - C > r \), then, because of the transformation uniformity condition, the cell can be pruned since there is no transformation in that cell that can map a point by more than \( r \) units, which is insufficient.
Section 8

Joint Detection and Tracking

8.1 Joint Detection

The method to detect an articulation is a little more elaborate than the one illustrated by figure 4.4. Simply taking the residual of the matching process as the result of the detection might be heavily corrupted by unmatched features that are unrelated to the articulation. To avoid this, the method illustrated in figure 8.1 is used instead. This method goes as follow.

1. Find a transformation $T$ that matches a part of $M$ with a part of $I$ a) and b).

2. The features that are exactly matched are removed and those unmatched are put aside c).

3. From b) the features that were almost matched are labeled d).

4. From c), a transformation $U$ is found such that it aligns the remains of $M$ and $I$. The features that are exactly matched are removed e) (and if the process would have more than one joint those unmatched would be put aside, just like step 2, and recursively processed the same way).

5. From e) the features that were almost matched are labeled f).

6. Taking the logical AND between the labeled features in d) and f) gives the detected joint without any false positives.
Figure 8.1: Algorithm for the detection of articulation.
The improvement of this method is that it generates less false positives. False positives are still possible, but are less likely to happen in a simple kinematics chain.

8.2 Joint Correspondence and Tracking

The solution to the problem of establishing feature correspondence is the ability to tell what went where. Typically, this is done by using a set of primitives in two adjacent frames and finding the relation that maximizes a similarity measure. Up to this stage of processing, the features available are fragments of edges representing the contours of articulated regions. The example of figure 8.1 g) shows nice connected contours of the joint, but in practice only a few unconnected points are likely to be detected.

So far, the only things that can be observed are chunks of edges moving from one place to another. To be able to understand the structure of the object, the system must be able to establish correspondence between the chunks of two consecutive frames. Usually, the choice of edge features makes correspondence unreliable since edges only provide one degree of freedom over small intervals [24]. Fortunately, this issue can be avoided by using the additional information provided by the geometric matching performed earlier.

The data available at time $t$ are features representing the contour of an articulation (those points will be called joint features from now on) and the transformations (say $T$ and $U$) that were used to find the articulation's adjacent limbs. Notice that with this information, the provenance of a joint feature cannot be exactly retraced by simply using the inverse of $T$ or $U^{-1}$. The reason for this is because the joint features are features that were almost matched by the transformations. Hence, there is no guarantee that transforming joint features at time $t$ by $T^{-1}$ or $U^{-1}$ would map them to actual edge features at time $t - 1$ (even though they might be close to one).

Instead, the transformations $T^{-1}$ and $U^{-1}$ can be used to decompose the object at time $t - 1$. Remember that what has been done so far is decomposing the object at time $t$ by finding $T$ and $U$. The properties of a metric allow going

\[ \text{Note that because of the logical AND, the two transformations } T \text{ and } U \text{ must map a point at time } t - 1 \text{ to the same point at time } t. \text{ Any of these transformations can be used as the one representing the transformation affecting the articulation since for features in the articulation area } T \text{ and } U \text{ are approximately equivalent.} \]
the other way without re-computing the Hausdorff distances: Inverse all the transformations, use the moving contours at time $t$ and decompose the object at time $t - 1$. This provides an additional set of observations at each time step, as shown in figure 8.2, at very little expense. One set of observations is obtained from the previous frame and the other from the next frame. Ideally, these two sets should be equivalent, however in practice this is seldom the case. To demonstrate how easy it is for the two sets to be different, imagine that a joint that is rotating between frames $t - 1$ and $t$ and then stops rotating between frames $t$ and $t + 1$. Between $t - 1$ and $t$ the rotation causes joint features to be extracted, but no feature is detected between frames $t$ and $t + 1$. However, the difference between the two sets is more often caused by differences of the extracted contours.

Having two sets of observations requires using a method to harmonize both sets in a consistent fashion and this is the goal of the next section.

### 8.3 Particle Propagation

Nowadays, probabilistic frameworks like Kalman and Bayesian filtering are widespread in all areas of computer vision. Their common strength is their ability to handle the uncertainty of the observation model and the dynamic model. For tracking purpose, Kalman and Bayesian frameworks combine the following ingredients:
• a Markov motion model $f_{t+1|t}(y_{t+1}|x_t)$: The probability of being in state $y$ at time $t + 1$ given the previous state $x$ at time $t$;

• a sensor likelihood model $f(z|x)$: The probability of making observation $z$ while in state $x$ (here the observation model is independent of time);

• the prior density $f_{prior}(x) = \int f(x|y)f_{prior}(y)dy$: The probability of simply being in state $x$ given all the possible previous states $y$.

For tracking applications, the posterior density $f_{t+1|t+1}(x_{t+1}|Z^{t+1})$ is determined by the recursive Bayes' equation

$$f_{t+1|t+1}(x_{t+1}|Z^{t+1}) = \frac{f(z_{t+1}|x_{t+1})\int f_{t+1|t}(x_{t+1}|x_t)f_{t|t}(x_t|Z^t)dx_t}{f(z_{t+1}|Z^t)}$$

in which $f_{t+1|t+1}(x_{t+1}|Z^{t+1})$ encapsulates all the information collected about the state variable up to time $t + 1$. To automatically extract the most likely state $x_{t+1}$ from the posterior density, a state estimator must be used (usually EAP or MAP).

When the densities are mixtures, using an analytical formulation of the posterior density is impractical because of the combinatorial explosion caused by the multiplication in Bayes’ rule. Therefore, methods were developed to circumvent this problem. One method that has been successful in recent years is the CONDENSATION algorithm [32].

Figure 8.3: Particle sampling: An approximation of a density by particles.
The CONDENSATION algorithm is based on particle sampling (and filtering), which approximates a distribution by a set of samples. Each sample is represented by a state and a weight, which corresponds to the probability of the state according to the probability distribution function. An example of particle sampling is shown in figure 8.3.

Just like densities, samples can be propagated through the recursive Bayes' rule by considering them individually as illustrated in figure 8.4. Starting at time $t$, the particles are sampled with replacement with a sampling probability proportional to their weight a). Then, each sample goes through the Markov prediction model that predicts in which state a sample should be at time $t + 1$ given its state at time $t$ b). Next, each predicted sample undergoes a Brownian motion to simulate the uncertainty of the prediction model c). Finally, the samples are modulated by the observation density in order to estimate their respective weight at time $t + 1$ d). This procedure is repeated.
at each frame such that the samples always represent the current knowledge about the state variable. Notice the strong interdependence between the observation and the motion model: if, for any reason, the sensor fails to make an observation where there are many samples (where it was predicted that there would be an observation) the weights of these samples will be small and they will not likely be sampled at the next time step.

When using Bayes' update rule, the two densities \( f_{t+1|t}(x_{t+1}|x_t) \) and \( f(z|x) \) need to be defined. Suppose that the state of a sample is given by its \((x, y)\) coordinates in the image. In our case, the Markov motion model can be replaced by the transformations found by the Hausdorff matching. As mentioned earlier, these transformations correspond approximately to the motion of joint features between frames. However, using deterministic transformations instead of a Markov prediction implies modifications to the CONDENSATION algorithm. With a Markov prediction model, it is always possible to predict what should be the next state of a sample; a sample goes from state \( x_{t-1} \) to state \( x_t \) to \( x_{t+1} \). However, in our problem, a transformation \( T \) determines how to go from state \( x_{t-1} \) to state \( x_t \) and another transformation \( V \) determines how to go from state \( x'_t \) to \( x_{t+1}' \) (figure 8.5). The problem consists of associating \( x_t \) with \( x'_t \).

The intuition of the proposed solution is to propagate a new set or samples at each time step. Take the example of figure 8.5 where two joints \( A \) and \( B \) are moving. The first step consists of representing the sets of joint features by sets of samples:

1. Each joint feature is modeled by a 2D Gaussian with its mean given by the coordinates of the feature and of fixed variance. Therefore, each set of joint features is represented by a mixture of Gaussians.

2. Sample the mixture with the Box-Muller algorithm [41].

Let \( A_t \) (white squares) and \( B_t \) (white circles) be two sets of samples representing respectively the joints \( A \) and \( B \) at time \( t \). \( A_t \) and \( B_t \) are both derived from joint features detected between time \( t - 1 \) and \( t \). Also, let \( S_t \) be a third set of samples (hashed squares) representing a joint at time \( t \). \( S_t \) is derived from joint features detected between time \( t \) and \( t + 1 \). At this stage

\[ ^2 \text{In fact, the transformation } T \text{ and } V \text{ does not even determine the mapping of states between time steps. They are only approximation of the motion of an articulation.} \]
it is not know which joint $S_t$ represents. If $S_t$ represents the joint $A$, then the samples of $A_t$ and $S_t$ should be combined to represent the joint $A$. The same argument is valuable for $S_t$ and $B_t$ if $S_t$ represents the joint $B$.

Therefore, the idea is to establish a correspondence between the sets of samples representing joints between frames $t - 1, t$ and those representing joints between frames $t, t+1$. The correspondence is established by successively multiplying the set $S_t$ with $A_t$ and $B_t$ and by computing the expectation of both resulting distributions. A correspondence can be established for the pairing that gives the highest expectation. This is analogous to maximizing Bayes' equation for multiple observation distributions. Imagine that the set $S_t$ represents the samples after the prediction step in the CONDENSATION algorithm. Also, imagine that there are two observation distributions available: $A_t$ and $B_t$ (suppose for now that $A_t$ and $B_t$ are actual probability distribution functions instead of sets of samples). The posterior distribution should reflect the observation distribution that is the most consistent with the predicted distribution. This consistency can be measured by a state estimator (EAP or MAP).

Once the correspondence is established between sets (i.e., $(A_t, S_t)$), the state of an articulation at time $t$ is represented by the product of its corresponding pair of sets (which has already been computed when establishing the correspondence). Furthermore, the estimated state of an articulation is given
by the expectation of that product (which has also already been computed when establishing the correspondence).

The same procedure is repeated at the next time step. Notice that there is no propagation of samples. At each time step, we already dispose of two sets of observations for each joint. One plays the role of the observation density, the other plays the role of the predicted density. The only thing that must be tracked in time is which joint is associated to which sets of samples (i.e. joint $A$ is associated with squares and joint $B$ is associated with circles).
Section 9

Results

Experiments were conducted on three different image sequences. Each of them tests one characteristic of the system. The next three sections illustrate results from the articulation detection on each of the sequences. The last section shows results from using the particle propagation algorithm for tracking. In all the experiments, the transformation group used for the matching was the affine motion. However, the parameters of the deformation matrix were constrained to values between $[0.75, 1.25]$.

9.1 Hand Gesture

Figure 9.1 shows snapshots of the original intensity sequence for the first sequence. The goal of this experiment is to show that the system can perform well even if the articulated part corresponds to a small fraction of the object.

In order to do this, the whole hand was slightly moving to the left such that the moving edges extraction would detect the contour of the hand. In order to detect an articulation, both of its adjacent limbs must be detected and thus, moving. For example, simply moving the thumb would result in only the contour of the thumb being extracted and matched between frames, without any chance of detecting the articulation. A solution to this would imply detecting the endpoints of each limbs and this problem has not been addressed.

The results in figure 9.2 illustrate that indeed the system is able to detect the source of the articulated motion. The joint features are the black pixels. Even if there are only few joint features this is enough to inform
Figure 9.1: Hand data.
where the joint is. Also, the small number of joint features should not be a surprise. Let's recall that joint features are features that were almost matched when matching both adjacent limbs. Therefore, the rule that determines what constitute an almost matched feature has an influence on the number of joint features. In these experiments, any feature that is within an 8-neighborhood of being matched is an almost matched feature.

Notice that the joint features can appear on both sides of the articulation. Ideally, both sides of the articulation would be detected at every frame, but this is seldom the case. Notice also that there are a few false positives, those are likely the result of the clutter in the area of the four fingers. However, contrary to real joint features, the false positives are sparse and inconsistent.

The thumb makes approximately 25% of the features in the images. The partial Hausdorff only considered 30% of inliers ($f_f = f_r = 0.3$).

### 9.2 Walking Human

Figure 9.3 shows the original intensity image of the second sequence. These images were also used for gait analysis in [43]. The goal of this sequence was to perform decomposition in cluttered objects.

The similarity between the background and the color of the shirt are responsible for the poor quality of the moving contours. Also, the junctions of the horizontal bars with the shirt create subjective contours. Yet, the joint detection performs fairly well. Except the outliers that appear in the feet, the joint features are concentrated around the hips and knees. A factor that is partly responsible for those outliers is the clutter caused by the endpoints and shadows on the ground. However, the quickness of the motion makes it almost impossible to detect the same joint in more than three consecutive frames. Therefore, using the tracking method for this sequence simply generate "flashing spots" corresponding to the short rotation time of the joints. The Hausdorff parameters were both 40% of inliers for the first match and 30% for the others.

The problem of false positives in the extremities is recurrent. The results for this sequence and for the next one indicate that most of the false positives appear in the extremities of the limbs. This is actually the consequence of a combination of details that were underestimated. The presence of clutter and edges tangential to the rotation in the extremities (i.e. shoes, fist, pants
Figure 9.5: Moving right arm data.
Figure 9.6: Moving left arm data.
Figure 9.7: Moving both arms data.
otherwise the affine matching is able to match the whole arm between frames.

The results for detecting the articulations for the sequences shown in figures 9.5, 9.6 and 9.7 are shown in figures 9.11, 9.12 and 9.13 respectively. These snapshots only show the joint features resulting from the decomposition of the object at time \( t \) from the object at time \( t - 6 \). Notice the few false positives around the hands in figure 9.13. Also notice that the system is robust to pieces of moving cloth around the waist and unrelated features from the head and shoulders. This robustness has the advantage that it does not require the subject to wear a skin-suit (or to be naked), as it is often the case with other systems.

The detections of the armpit in a few frames are actually false positives and should not be considered as detecting a bonus articulation.

Each execution of the Hausdorff distance with a single arm motion took approximately five minutes on an AMD Athlon 1.4GHz with 512MB of memory. When both arms are moving the execution time was between 12 and 15 minutes. The reason for this because the transformation uniformity condition that is necessary to compute the Hausdorff distance. Let's remember that the larger is the model, the finer is the rasterisation of the transformation space. Therefore, the computation time is bounded by the size of the model, not the number of features.

**Tracking**

An example of using the deterministic version of the CONDENSATION algorithm to update the “posterior” density is shown in figure 9.14. Figure a) shows samples representing the set of observation obtained between frames \( t - 6 \) and \( t \). Figure b) shows the set of samples obtained between frames \( t \) and \( t + 6 \). Finally, figure c) shows the updated set of samples. Notice that in a), only the lower part of the elbow is detected whereas in b) the upper part is also detected. The explanation for the presence of samples for the upper part of figure c) is because of the actual presence (though less dense) of samples in that area in b). These samples are not shown in order to avoid cluttering the figure. The standard deviation of the Gaussian modeling each point feature was 10 pixels. This value is purposely high to allow bridging the gap between joint features on both sides of the articulation.

Tracking was performed on the entire sequence. Results of the tracking
Figure 9.2: Hand joint detection.
Figure 9.3: Gait data from [43].
and shirt borders) is the cause for the high occurrence of false alarms in those areas. For example, the first match aligns the torso by translating the model a few pixels to the left. Doing so also aligns other parts, like the front of a foot with the back of the same shoe. However, the relatively high density of the clutter in those areas causes many features to be almost matched and those features are then likely to be almost matched again when the limb that they belong to is segmented.

9.3 Moving Arms

The purpose of this sequence is to show one of the strongest advantages of using a bottom-up approach. The sequence starts with a person moving his right arm. As it should, the system decomposes the forearm and the upper arm and detect joint features around the elbow. To show how adaptable the system can be, the right arm stops moving and the left arm starts doing the same motion. The system stops detecting the right elbow, as no more motion is perceived, and starts detecting the left one instead. Finally, both arms are moved simultaneously and the system is able to detect both elbows at the same time.

The initial data for the moving arms are shown in figures 9.5 (left), 9.6 (right) and figure 9.7.

An example of piecewise segmentation for the right arm is shown in figure 9.8. The figure 9.8 a) illustrates the contours of the model \( M \) and b) the image \( I \). Figure c) illustrates the matched features obtained by aligning the forearm and d) illustrates the almost matched features of the forearm. Figures 9.8 e) and f) illustrate the unmatched features of the model and the target respectively. Those remaining features are used for the matching of the upper arm. Figure 9.9 a) shows the matched upper arm and figure 9.9 b) shows the features almost matched. The joint features are the result of the logical AND of figure 9.8 d) and 9.9 b) and are shown in figure 9.10.

Notice the large difference in the pose of the arm between the two frames. For this sequence, six frames were interleaved between the model \( M \) and the image \( I \). That is, the frames \( t \) and \( t + 6 \) were used for the decomposition. The reason for this is the combination of the slow motion of the arm and the frame rate of 30fps resulted in very little deformation between the frames. To achieve better results, a fair amount of deformation must be perceived,
Figure 9.4: Gait joint detection.
Figure 9.8: Forearm decomposition.
Figure 9.9: Upper arm decomposition.

Figure 9.10: Elbow detection.
Figure 9.11: Right elbow detection.
Figure 9.12: Left elbow detection.
Figure 9.13: Both elbows detection.
for the three parts of the sequence are shown in figure 9.16, 9.15 and 9.17. The snapshots show the updated "posterior" distribution represented by the samples.
Figure 9.15: Right elbow tracking.
Figure 9.16: Left elbow tracking.
Figure 9.17: Both elbows tracking.
Section 10

Future Work

Perhaps the simplest improvement that could be made on the system would be to track individual components once they are decomposed. The reader should have noticed that the system performs a piecewise decomposition at each frame and this is not always necessary. Performance might be improved by tracking individual limbs once the system has already found them. However, a decomposition must occur in the case where a previously undetected joint starts rotating.

A problem often encountered with matching edges under affine transformation is when the edges of the model are collinear. Such is the case when tracking human limbs. In those circumstances, the model often offers a best match when it is simply rotated in or out of the image plane and shrunk to its acceptable minimum such that it can easily match a subpart of the target that it is supposed to match. A possible solution to this is to combine edges with different features such as corners or SIFT features [45] to reduce the “weight” of collinear edges in the matching.

Obviously, the detection of joint features depends on the presence of edge features in the jointed area. Another way to generate an observation is to infer the location of an articulation given the transformation of its adjacent limbs. Given two transformations $T$ and $U$ in the image coordinate system, the goal is to find the origin of the coordinate system centered at the articulation as shown in figure 10.1. The solution to this problem can be expressed as finding the coordinate system such that the origin of the transformation is centered on a joint. For example, imagine that the geometric matching matches the limb $A$ with a transformation $T$ and the limb $B$ with the transformation $U$, the problem consist of finding for the coordinate system centered at $O = (O_x, O_y)$
that is centered on the joint.

Let the angles of rotation $\theta$ and $\alpha$ be in the $(x, y)$ coordinate system and $\theta'$ and $\alpha'$ be in the $(u, v)$ coordinate system. Then for features $(a_x, a_y) \in A$, $(b_x, b_y) \in B$ in the $(x, y)$ coordinate system and $(a_u, a_v) \in A$, $(b_u, b_v) \in B$ in the $(u, v)$ coordinate system we have

\[
\begin{bmatrix}
\cos(\theta) & -\sin(\theta) & T_x \\
\sin(\theta) & \cos(\theta) & T_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
a_x \\
a_y \\
1
\end{bmatrix}
= 
\begin{bmatrix}
\cos(\theta') & -\sin(\theta') & \delta O_x \\
\sin(\theta') & \cos(\theta') & \delta O_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
a_u \\
a_v \\
1
\end{bmatrix}
\]

\[
\begin{bmatrix}
\cos(\alpha) & -\sin(\alpha) & U_x \\
\sin(\alpha) & \cos(\alpha) & U_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
b_x \\
b_y \\
1
\end{bmatrix}
= 
\begin{bmatrix}
\cos(\alpha') & -\sin(\alpha') & \delta O_x \\
\sin(\alpha') & \cos(\alpha') & \delta O_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
b_u \\
b_v \\
1
\end{bmatrix}
\]

which is underdetermined since $O$ is unknown. However at $O$, we have
\[
\begin{bmatrix}
\cos(\theta) & -\sin(\theta) & T_x \\
\sin(\theta) & \cos(\theta) & T_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
O_x \\
O_y \\
1
\end{bmatrix} =
\begin{bmatrix}
O_x \\
O_y \\
1
\end{bmatrix} +
\begin{bmatrix}
\delta O_x \\
\delta O_y \\
0
\end{bmatrix}
\]

\[
\begin{bmatrix}
\cos(\theta) & -\sin(\theta) & U_x \\
\sin(\theta) & \cos(\theta) & U_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
O_x \\
O_y \\
1
\end{bmatrix} =
\begin{bmatrix}
O_x \\
O_y \\
1
\end{bmatrix} +
\begin{bmatrix}
\delta O_x \\
\delta O_y \\
0
\end{bmatrix}
\]

Therefore, the solution to locating the joint can be found by solving these equations. Notice that this method would create some problems if the kinematic chain would be composed of more than two limbs. In which case each transformation must be paired appropriately with the transformation of an adjacent limb.
Section 11

Conclusion

Despite the fact that the theory of structure from motion has been established almost a quarter of a century ago and has been successfully implemented in various ways, no such attempt has been made for articulated objects.

Many systems that combine top-down processing with a priori knowledge have already been developed. The number of assumptions that these systems require makes them ill-suited for tracking arbitrary articulated objects.

This thesis introduced a novel way of analyzing articulated motion. Its intention was to demonstrate that complex motion could be analyzed without relying on the conventional assistance of shape models or any form of knowledge.

The first piece of the puzzle was to find a suitable generic representation. A representation based on the joint configuration was proposed. This representation has the advantages of being generic, bottom-up extractable from raw images and is sound from a psychological point of view.

The other piece of the puzzle was to avoid the issues of top-down information flow. Therefore a new bottom-up method was developed to detect and track the joints. The first step involves an efficient moving contours extraction technique that uses a combination of high-order moments and interframe difference. Then, the partial Hausdorff distance is used to recursively decompose the moving contours into rigid components. From this decomposition, the joints are extracted by detecting the overlapping segment of edges between adjacent components. The method also generates a second set of observations for each frame by decomposing the moving contours at the previous time step. These sets of observation are fused together by using a modified version of the
CONDESATION algorithm.

The results showed that the method could indeed track multiple joints without requiring any knowledge about the object. The results concretized the advantages of avoiding a priori knowledge and top-down process by revealing the generality and adaptability of the system. It was expected that the system would be less accurate than knowledge-based systems. However, tasks like recovering the structure of an articulated object do not require a much higher level of precision.
Bibliography


76


