THESIS

MOTION SEGMENTATION FOR FEATURE ASSOCIATION

Submitted by

Weston Pace

Department of Computer Science

In partial fulfillment of the requirements

for the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2010

ABSTRACT

MOTION SEGMENTATION FOR FEATURE ASSOCIATION

In a feature based system physical objects are represented as spatial groups of features. Systems which hope to operate on objects must make associations between features that belong on the same physical object. This paper segments interest points in individual frames of an image sequence using motion models based on image transformations. Experiments evaluate the associations made by these segments against ground truth data. We give an improved version of the existing algorithm which can lead to easier threshold selection in some systems although the ideal threshold is shown to depend on the goal of the segmentation. Lastly we show that the underlying motion of the object is not the only factor in determining the performance of the segmentation.

ACKNOWLEDGEMENTS

This thesis has been a journey of considerable learning and effort. It is certainly a journey I would not have even begun, much less finished, were it not for the support of numerous family, friends, teachers, and mentors.

First, I'd like to thank my wife Christie Pace, who stuck with me from the start to the finish. I should point out that stress, exhaustion, and worry do not make me a pleasant person to live with. I would have given up long ago were it not for her encouragement. I want to also thank the rest of my family, especially my parents Rod and Nina pace. They spent countless years raising me and are the foundation of who I am today. They gracefully accepted my decision to take a break from this thesis yet never stopped encouraging me to finish it.

While my family provided emotional and spiritual support I want to also thank those who provided educational support. First and foremost I'd like to thank my advisor, Dr. Bruce Draper, for encouraging me to begin research in the first place. In addition, he showed how to do research correctly and this thesis is based on countless hours of his guidance and advice. I'd also like to thank the rest of my committee, Dr. Ross Beveridge and Dr. Stephen Hayne for their support of this work.

I also want to thank Elaine Regelson who has no equal when it comes to giving smart freshmen motivation and keeping them out of trouble. My thanks goes out to the rest of the Colorado State University teaching staff, in particular, Dr. Wim Bohm, Dr. Ross McConnell and Dr. Sanjay Rajopadhye.

I am extremely grateful for Robert Weant and Alan Copeland who helped label the ground truth data used in this work.

Finally, I want to thank my friends in the trenches, Nick Parrish, Stephen DiBenedetto and Douglas Moore, as well as the entire CSU robot club. Together they provided emotional support through my entire time at CSU. I would also like to thank Lutheran Campus Ministries for providing spiritual support as I worked on this thesis.

DEDICATION

This thesis is dedicated to my current and future family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Automatic analysis of image sequences is a complex task that quickly becomes computationally intractable. One popular approach to this problem is to locate key features[Low04] in an image to reduce the amount of information that needs to be analyzed. Unfortunately, these methods create a new challenge - piecing the image back together. This thesis studies the use of motion information extracted from tracked image features to create robust and meaningful associations between features.

Recent works have employed a random search for motion using models based on image transformations. That particular approach is studied in more detail in this work. Parameters in the algorithm are studied and a new measure is presented which leads to a more effective threshold selection. It is shown that correct associations can be made through motion information although most of the associations are formed between background points. Furthermore it is demonstrated that this distinction is not because background objects are less error prone or somehow more describable by motion. These findings provide a base for the study of motion for feature grouping.

1.1 Feature Grouping

Discovering relationships among features is a critical component for any feature based vision system intended to accomplish complex tasks. These relationships can be used to create percepts. Percepts are a representation of associations between physical structures. Percepts provide the user with predictive information that can help other tasks such as feature detection, anomaly detection, occlusion recognition, and location recognition. Percepts can also be used as building blocks for higher level problems including object recognition, visual information retrieval, and complex navigation. Percepts can be formed from features through feature associations

Associations allow us to express a unique relationship amongst a group of features. Spatial associations are associations between features in the same image while temporal associations associate features in different images. A common type of temporal association is a feature descriptor. Feature descriptors categorize features based on the structure of the region surrounding the feature. Feature descriptors typically represent a particular viewpoint of a physical structure and so a percept can be expressed as an association between feature descriptors. For example, a face percept might consist of feature descriptors for the eye, nose, and mouth.

Since an association between features could indicate an association between the underlying feature descriptor one technique for grouping feature descriptors is to find spatial associations. Several algorithms already exist for detecting spatial associations. Some examples are the use of shape models to find common constellations of features[FFFP04], color and texture based models which group features through traditional segmentation algorithms[SWRC06], and cooccurrence models which group features that occur at the same point in time[SZ03].

One characteristic of the previous methods is that they all operate on a snapshot or collection of snapshots, images that need no explicit relationship to each other. In reality, images are often given in image sequences provided from a camera or movie. In such situations there exists a motion relationship between the images that can be used as an additional source of information. This work examines the effectiveness of that motion information for forming spatial feature associations.

1.2 Image Sequence Analysis

There is a large body of work which studies the extraction of motion-related information from image sequences. Some typical motion tasks are tracking the movement of objects in order to predict their future location[LK81], identifying anomalous motion patterns for intrusion detection[CBK09], and attempting to extract three dimensional structure from 2D representations of objects[Ull79].

These tasks work by creating some kind of model that describes the underlying motion. There are three general categories of motion models[LGF10]. Trajectory based models attempt to describe the motion of a single point through time. Point based trackers with linear filters are an example of a trajectory model. Transformation based models describe the motion of all points at a single point in time. Transformations are often used in tasks which attempt to calculate structure from motion. The third model group is flow based models which describe the motion of all points over time. Flow models are complex and typically nonlinear unless certain constraints are made.

With the rising popularity of feature extraction there now exist a number of techniques that fit tracked features into these models. Feature based motion models are used to extract independently moving objects, or clusters of points, from video sequences and assign a motion model to each object. These groups can provide us with a motion based segmentation of each frame of a video sequence. This segmentation groups features which move similarly, independent of the camera motion.

1.3 Percepts from Motion

The purpose of this work is to study the suitability of image sequence analysis for the task of percept formation. By tracking features, feature velocity information is obtained. Transformation based models are then used to create feature associations by grouping features that move together at a given instance in time. This should lead to the creation of robust percepts unique from those created by other methods.

In this work an extension to Colorado State University's SeeAsYou system is described that extracts motion associations. The algorithms and approaches taken by the system are examined as well as the parameters involved in the process. The extension is then tested through a series of experiments to determine the accuracy of the associations based on ground truth labels. Finally, the results of these experiments are presented and discussed.

Chapter 2

Background

2.1 Feature Detection

Feature detection is a first step in many computer vision systems. The goal is to extract features in such a way that these features form a more stable and more compact representation of the information contained in the original image. Typical features have some sort of central point, which is referred to as the interest point. In addition, features are also represented by a region of pixels around that point, called the feature region. Many types of features have appeared over the last few decades to fill the needs of a diverse set of tasks.

One of the oldest and best studied features is the Harris corner[HS88]. This feature was originally designed to be a stable feature for tracking. The Harris corner selects interest points such that the feature region has edges in two orthogonal directions. One of the reasons the Harris corner has become such a popular feature is that it tends to be repeatable. That is, it will be detected despite minor changes in rotation, scale, illumination, and image noise.[SMB00] This repeatability criteria has led to the development of more sophisticated features.

One such feature is to search for peaks in an image's response to a difference-of-gaussians filter. This tends to select interest points whose regions contain blob like patterns. This feature was originally developed by Lowe[Low04] and tends to perform near state of the art for repeatability[SMB00]. Typically the feature is used along with an image pyramid to enhance the runtime of the algorithm.

An image pyramid is a series of images where each image has high frequency information above a certain cutoff removed. This frequency cutoff point is typically referred to as the scale of an image. The larger the scale of an image the lower the frequency cutoff. This pyramid is typically constructed through both applications of a Gaussian filter and subsampling of the image. The subsampling stages remove the total number of pixels in an image. As a result such pyramids can allow for a logarithmic reduction in the number of pixel comparisons required to detect features at different scales.

2.1.1 Feature Descriptors

Another development in feature based vision systems is that of feature descriptors. A feature descriptor is usually a classification based on the feature region. Each feature descriptor represents a wide range of possible feature regions that meet some set of criteria. Ideally a descriptor should be a viewpoint dependent representation of some physical object. One of the most well known examples of such a descriptor is the SIFT descriptor[Low04]. The typical goal of such descriptors is to create a transformation invariant description of a feature region.

Transformation invariant descriptors lead to descriptors which perform well under a similar repeatability criteria. An effective feature descriptor should classify a feature with the same descriptor despite changes in the perception of the feature itself. These descriptors are useful because they create associations between features in different images.

The performance of a feature descriptor depends not only on the algorithm used to classify the feature region but also on the algorithm used to detect the feature region. This has led to a new criteria for feature extraction. New features are emerging that are based on their tendency to produce affine covariant regions, regions that change covariantly with affine transformations. Such regions are shown to lead to affine invariant descriptors. For a thorough review of these regions we refer the reader to [MTS+05]

2.1.2 Feature Association

Feature descriptors segment the space of all possible features. This allows for associations to be made between features that are given the same descriptor. Associations between features can be grouped into two broad categories, temporal associations and spatial associations. A spatial association is an association between two features in the same image. A temporal association is an association between a feature in two different images.

The associations made by feature descriptors are designed to be temporal. Since a feature represents some aspect of the physical world then every image of that physical object should contain the feature and all of these features should be associated by a feature descriptor. By contrast, a feature descriptor should be duplicated in an image only if the underlying piece of the real world was also duplicated.

Feature associations are critical because many human and robot tasks operate on physical objects. In a system that operates on features, physical objects are typically represented by spatial groups of features. For this reason, one of the most important tasks for a vision system that hopes to operate on objects is to form these spatial groups.

One way to create spatial associations between features is by using associations between feature descriptors. If you know that two descriptors X and Y are associated and you detect features x and y such that the descriptor of x is X and the descriptor of y is Y then you can create a spatial association between x and y. An association between feature descriptors is often referred to as a percept.

As an example consider a system that is trying to recognize trees. Let the system have a feature descriptor which categorizes features into a set of descriptors, two of which are leaf and bark. The spatial percept formed by the leaf descriptor will capture the top of the tree but not the bottom. The spatial percept formed by the bark descriptor also fails to capture the whole tree. If we know that the leaf descriptor and the bark descriptor are associated through a tree percept then we can form the leaf or bark spatial group and identify trees in an image.

Co-Occurrence is one of the most common forms of percept formation and is used in a work by Sivic and Zisserman[SZ03] to learn about objects in a dataset. In this work Sivic uses multiple approaches to detect affine covariant regions. Each region is then assigned a SIFT descriptor. Sivic uses co-occurrence to associate all descriptors that tend to occur in the same frame. Once this is done Sivic uses these percepts to support object retrieval tasks. The goal of this thesis is to study the use of motion segmentation as a source of spatial associations. The assumption is that if two features move in the same way then those features should be associated. These associations could be used by themselves but ideally they would be used in conjunction with other forms of association such as feature descriptors. This would allow for percepts to be created more accurately than through simple co-occurrence.

2.2 Tracking

Tracking is an old problem for which many approaches have been developed. Images are typically 2D representations of a 3D space. As objects move through 3D space their associated 2D projections undergo various transformations. There are numerous factors that make tracking a difficult task. These include camera noise, nonrigid motion, occlusions, and lighting changes.

Tracking algorithms can be categorized by the underlying model they use to represent a tracked object. Since we are dealing with features in an image we are interested in algorithms that can track features, either by tracking interest points or by tracking feature regions. For a review of the general tracking problem, including other types of algorithms which track objects using more sophisticated models we refer readers to [YJS06].

The Lucas-Kanade tracking algorithm is a well studied approach to tracking. This algorithm is ideal for this thesis because it operates on features, is well-established, and is capable of running in real time. The algorithm is based on a phd dissertation where Lucas developed an algorithm for image alignment[Luc84]. Given a source image, a destination image, and a parameterized motion model the algorithm will calculate the set of parameters that best describe the transformation mapping the source to the destination.

To track points Lucas and Kanade [LK81] extended this algorithm to take a window in a source image, along with a destination image and calculate the transformation undergone by that window. Applying this transformation to the window boundaries will give the new location of the window. Tracking can then be thought of as simply properly registering this window through each frame of an image sequence. One drawback to the algorithm is it becomes difficult to track motions that are much greater than the window size. In addition, the running time of this algorithm is proportional to the window size so there is significant motivation to keep the window size small.

To handle larger motions one can further extend this algorithm with a multiscale image pyramid[Bou00]. This pyramid is much like the one described in the feature detection section. The new algorithm iterates through each layer of scale to find a transformation. The tracking window remains the same size at each layer while the image size shrinks. As a result the tracking window represents a greater portion of the image as the scale of the image increases. As an added boost to runtime we can reuse the image pyramid computed for feature detection. Tracking forms the basis for more complex motion tasks which rely on point correspondences.

2.3 Structure from Motion

Computing 3d structure from camera motion is a nearly 30 year old problem. The task is to compute the 3d geometry of a physical scene or an object in the scene given a sequence of 2d images of the actual scene. This requires that either the camera, or the object to be measured, be in motion. In the general case there are situations that can arise which will lead to an infinite number of correct solutions. In 1979 Ullman proved the existence of a solution given three views of four non-coplanar points in certain scenarios[Ull79]. Ullman showed that there would always be a unique solution provided the underlying scene was rigid.

Unfortunately, research has shown that this problem is ill defined when the image is noisy[RA80]. Consider a large object in the distance. Noise that results in shifts of a few pixels can correspond to large changes in scene geometry. Much work has been done since Ullman's result to try and establish robust solutions to this problem. In 1992 Tomasi and Kanade developed an algorithm based on the singular value decomposition[TK92] which computes the scene shape and camera motion without computing an intermediate representation of the depth which avoids the problem of noisy distant objects. This approach can also use all tracked points to hopefully reduce the effect of outliers. One drawback is the approach assumes that nothing in the scene is moving independently of the camera.

To deal with the multiple motion scenario Torr developed an algorithm based on RANSAC[FB81] which used local features to solve the structure from motion problem[TZ00]. RANSAC is a model selection algorithm described in more detail in chapter 3. RANSAC searches through random subsets of the available points in order to come up with a model that describes as much of the input as possible. This algorithm has been shown to perform well even in the presence of independently moving objects.

2.4 Multibody Structure from Motion

One of the side effects of using RANSAC to calculate the camera motion is the production of a set of outlier points which can be assumed to belong to one or more of the objects moving independently of the background. In his phd work, Torr studied the outliers detected in a structure from motion task and took advantage of this fact[Tor95]. Torr recursively ran the RANSAC algorithm on the outliers to identify the motion of the independently moving objects in addition to the larger scene motion.

Fitzgibbon and Zisserman used this approach on the original structure from motion task[FZ00]. They found that sequences with multiple moving objects often allowed for a more precise computation than static-scene based reconstructions. They were even able to develop structure from motion in some cases where static reconstruction was underconstrained.

In 2005, Vidal took an algebraic approach to solve for the motion of all independently moving objects at once[VMSS05]. Unlike the RANSAC approach Vidal's approach is guaranteed to find the correct solution. While promising, the approach currently requires factoring a polynomial with degree equal to the number of independent motions.

2.5 Motion Segmentation for Percept Learning

In 2004, Rothganger et al. adapted the idea of motion segmentation to create 3D object models from a video sequence[RLSP07]. Their work first detects Harris and Hessian interest points and affine covariant regions are defined around these interest points. The regions are then tracked with a modified version of the Lucas-Kanade tracking algorithm. This algorithm first predicts a location with standard Lucas-Kanade techniques and then refines the prediction with non-linear least squares. These tracks are then grouped using the process described by Torr. Each group of tracks is considered to be an object and a corresponding 3D model for the object is created. These objects are then used for object detection and shot matching in novel images.

A number of techniques are used to reduce the effect of noise. First, once all tracks have been extracted from the video sequence the frame f_{max} containing the largest set of *stable* tracks is found. Given a frame f a *stable* track is defined as a track that appears in frames $[f, f + \omega)$ where ω is a parameter. This set of tracks contained in f_{max} is then given as input to the previously mentioned RANSAC algorithm to find a set of stable and connected tracks, C. The set C is then used to form an object model. Finally, This object model is then grown by adding other tracks from anywhere in the sequence that fit the model. If the formed object does not contain a sufficient number of tracks then the object is abandoned.

By starting with the largest group of motion, this procedure attempts to create the best possible starting model of an object. Many tracks from other scenes are then eliminated from further motion segmentation by the object model fitting stage. Finally, using only stable tracks and removing models that are not present in enough frames eliminates noisy or weak motions.

In 2005, Sivic and Zisserman used a similar approach but worked on a slightly different task[SSZ05]. Their work analyzed a video sequence with the goal of indexing the sequence for later searching. Objects were created from features in an image that were consistently segmented together. This allowed for users to submit a query (a hand selected object in any image of the sequence) and in realtime retrieve other images in the sequence containing the same object. In particular, users could find different views of the object. This includes view changes due to movement in the scene, such as the front and back of a van that drove past the camera, and view changes due to object deformation, such as the mouth of a talking person.

Sivic's procedure is similar to Rothganger's in that it also uses a combination of Harris and Hessian points with affine covariant regions. Sivic then uses a novel tracking method based on a simple matching combined with a track repair algorithm which attempts to fix track gaps with surrounding motion. Sivic uses Torr's RANSAC procedure with homographies to find motion segmentations.

Again, care is taken to handle noise. Sivic's algorithm finds a motion segmentation for every triplet of frames. Sivic adapts the RANSAC algorithm to operate on a frame triplet by passing it three pairs of frames for the triplet, one pair for each possible combination. Once this is done a track-by-track association table is created. Two tracks are said to be associated in a pair of frames if they are segmented together in that pair by the RANSAC procedure. The procedure then applies standard text retrieval methods on the association table to form object groups. Sivic describes the purpose of this step as removing outliers and noise.

There are a number of similarities between Sivic's method and Rothganger's method. Both of them use RANSAC with projective transformations to find motion based associations. Both methods are runtime intensive and use fairly sophisticated tracking mechanisms. Finally, both methods use some kind of global information or repeated trials to filter out the noise produced by the motion segmentation algorithm.

In this thesis we analyze the motion segmentation step used by Sivic and Rothganger. In contrast to their methods we use much simpler interest points and tracking algorithms. In addition to projective transformations we experiment with a set of more primitive motion models. Finally, we look at the raw associations produced by the segmentation and evaluate their accuracy with respect to ground truth. We attempt to identify which variables affect the performance of the segmentation and how they do so.

2.6 SeeAsYou

This work is based on a computer vision system developed at Colorado State University named SeeAsYou. The system supports a dataflow architecture represented by modules and buffers. Each module takes input from zero or more modules, performs some analysis on the input, and then outputs the result of that analysis to zero or more buffers. For a detailed description of the SeeAsYou system we refer the reader to [DT10]

This thesis makes use of several existing modules of the SeeAsYou system. The eyes module reads images from a camera or file and outputs those images in a format familiar to the system. The salience module detects blob shaped features in the image with a difference of Gaussians filter using a multiscale image pyramid. The salience module outputs the interest point location and a circular region surrounding these features.

This work further extends the SeeAsYou system by adding two new modules. The first module added is a simple tracker designed to provide point correspondences between frames. The second module is an implementation of the RANSAC based motion segmentation algorithm previously described. The output segmentation is used to create feature associations. These associations would then be sent to a module which creates percepts however our evaluation happens before that step.

Chapter 3

Methods

In this chapter the various algorithms used in this work are introduced. In addition, any parameters involved are discussed and the values of those parameters are presented.

3.1 Tracking

For this study a tracking module was created in the SeeAsYou system. This module attempts to track a feature through a sequence of images. If a feature is tracked successfully then the module marks the feature as tracked and augments the feature with information describing its motion. The module represents the path a feature takes through a very simple model. Each tracked feature is given a starting position and a current position. The motion of the feature is then a vector from the source to the destination.

The tracking module operates on two basic units of data. The first unit is a set of features detected in the current image. Some of the features may then be given motion information and passed on. The second piece of data is a set of targets which represent the current motion tracks in progress. Each target has a template which describes what the feature should look like. Each target also has a first-in-first-out queue of T_L positions, $\{P_1, P_2, ..., P_{T_L}\}$ which gives the position of the target in the T_L most recent frames. P_1 represents the most recent position and P_{T_L} represents the source position (although not necessarily the first detected position) of the target. The motion models used in this work are transformations between frames and so $T_L = 2$.

3.1.1 Feature Location Prediction

The first two steps of tracking predict and confirm the position of a feature in a new frame given the position of that feature in the previous frame. In these two sections the known old position of the feature will be referred to as the source position. The unknown position of the feature will be referred to as the destination position.

The location prediction step attempts to find the most likely destination position of a feature given the source position. To accomplish this we chose to use a pyramidal implementation of the Lucas-Kanade feature tracker described in chapter 2. The code for this algorithm was provided by OpenCV. The exact implementation was based on work by Bougeuet.[Bou00]

The algorithm has three parameters. The first is the size of the window placed around the feature point. Ideally this would be based on the size of the feature as given by the salience module. For simplicity a 5x5 window was chosen for all features. The next two parameters determine the method of convergence for the algorithm. Once again, for simplicity, the algorithm was set to iterate twenty times and then finish.

The performance of the prediction algorithm was verified subjectively through a series of tests validating known input. Since this algorithm was not a primary concern of this work we did not perform any extensive testing on the performance or accuracy of the algorithm. The parameters described above were found to give adequate performance and were never modified in any of the experiments described in this paper.

3.1.2 Feature Location Confirmation

In order for a feature to be tracked between two frames the tracking module requires that the feature be detected in both the source frame and the destination frame. This is not strictly necessary since it is possible the feature might not be detected in the destination image but still tracked. However, features that were ”lost” by the salience system were more likely to be noise and so they were discarded. The module also requires that the features move as predicted by the Lucas-Kanade algorithm.

These requirements are enforced by the location confirmation step of our procedure. Once a feature has been given a predicted location the module searches through all features within a given search radius of the predicted location. To optimize this search slightly a spatial grid of all extrema detected in the destination image is created before any searching is done.

This step has two parameters. The first is the search radius which determines the distance (in pixels) that a feature can deviate from its predicted position. A search radius of 20 pixels was chosen for this work. The second parameter determines how close a feature has to match the template of a target to be considered a match. The features are compared through correlation and this work requires a correlation of 0.75 for a match.

3.2 Motion Segmentation

The final step of the system as far as this work is concerned is motion segmentation. The goal at this step is to group point correspondences that move together. This provides later parts of the system with an association between the features that contribute to these point correspondences. Once we have these feature associations we can use techniques similar to those used in Sivic et al[SZ03] in future modules in the SeeAsYou system to form object percepts.

In the following sections methods used to generate this motion segmentation are described. First, the interpolation process which actually generates motion models from points is described. In addition we explain how point correspondences are matched to models. Next, the RANSAC (Random sampling and consensus) model selection and evaluation process is presented. Finally an alternate error measure motivated by patterns observed in our experiments is presented. In the following sections we will refer to the input set of tracked points as P. Each point $p \in P$ represents the movement from a source point $p_s = (x, y)$ to a destination point $p_d = (u, v)$.

3.2.1 Model interpolation

The RANSAC algorithm repeatedly generates sets of tracked points $R \subset P = \{p_1, ..., p_k\}$. For each set R a motion model A is generated describing R. The process to generate A depends on the motion model desired.

Magnitude: R contains a single tracked point p. This model is the only model which is not strictly a transformation. The model generated represents motion in any direction such that the magnitude of motion is equal to $d = |p_d - p_s|$.

Translation: R contains a single tracked point p. The transformation generated is:

$$\begin{matrix} 1 & 0 & u-x \\ 0 & 1 & v-y \\ 0 & 0 & 1 \end{matrix}$$

Affine: R contains three tracked points. The general affine transformation can be represented by 6 parameters:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

To find the parameters one must solve for A such that A is of the form just described and $Ap_s = p_d$ for each of the tracked points. Given this criteria a system of 6 linear equations is generated:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix}$$

Solving this system gives the values of each of the affine parameters.

Projective: R contains four tracked points. The general projective transformation is:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Similar to the affine case one can construct 8 linear equations from the 4 tracked points:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1v_1 & -y_1v_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2u_2 & -y_2u_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2v_2 & -y_2v_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3u_3 & -y_3u_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3v_3 & -y_3v_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4u_4 & -y_4u_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4v_4 & -y_4v_4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{bmatrix}$$

3.2.2 RANSAC model selection

RANSAC[FB81] is a statistically motivated algorithm used to search for a model describing the input. The process has been shown in a number of situations to robustly select the correct model despite significant outliers in the input. RANSAC can be broken down into two stages. The first stage is to randomly sample the input in order to create a model. The second stage evaluates the generated model by measuring consensus amongst the input.

The RANSAC algorithm has four main parameters. The first parameter, k, is model dependent and represents the number of input items required to generate a model. The second parameter, w, represents the percentage of inliers in the input set. Given w, p is defined as the probability that the algorithm will select k inlier points at least once during the selection process. Finally, t is a threshold used for determining consensus. The parameter t is often referred to as the inlier threshold.

In our experiment w was fixed at 0.3. This number was selected because the experiments will look for up to 3 independently moving objects. In the worst case each of these objects would be of equal size and after the background is removed the true value of w would be $\frac{1}{3}$. By selecting 0.3 we err on the side of caution. In reality the true number of objects is not known and this system would not perform well on more than 3 objects given the fixed w value. The system also fixes p at 0.95 as this is a typical value.

The experiment used recursive iterations of RANSAC to segment the input. In each stage of the recursive algorithm the goal is to randomly select tracked points from the input set of tracked points $P = \{p_1, ..., p_r\}$ to generate motion models. The number of tracked points required (k) depends on the motion model. Table 3.1 gives the number of points required to generate each of the motion models used in this paper. The algorithm should generate enough models, n, that it can satisfy the requirements imposed by p. Provided the input points are selected independently then $\hat{n} = n$ can be calculated from p by:

Motion model	k
Magnitude	1
Translation	1
Affine	3
Projective	4

Figure 3.1: k values

$$\hat{n} = n = \frac{\log(1-p)}{\log(1-w^k)}$$

After calculating $\hat{n}$ the system would then randomly select $\hat{n}$ combinations of k points from P. In this work these combinations must be chosen without replacement and so the points are not selected independently. This means that $\hat{n}$ cannot be used for this work. In reality the true value of n is not largely different than $\hat{n}$ and this work sets $n = \min(\hat{n}, \binom{|P|}{k})$. For each of the n combinations the system interpolates a motion model A using the previously described methods resulting in the set $\mathbf{A} = \{A_1, ..., A_n\}$.

Once a model has been generated the algorithm must measure the quality of the model. To do this RANSAC associates an error measure, $e_{p|A}$ for a point p given a model A. If A is a transformation then $e_{p|A}$ is the projection error of the transformation. Given that a tracked point p represents a movement from a source point p_s to a destination point p_d the projection error is:

$$e_{p|A} = |Ap_s - p_d|$$

The magnitude motion model used in this experiment is not a transformation and $e_{p|A}$ must be calculated differently. Given that a magnitude motion model representing motions with magnitude equal to d:

$$e_{p|A} = |d - |p_d - p_s||$$

For all models, if $e_{p|A} < t$ then our point is considered to be an inlier with respect to A. The consensus score for A referred to as S_A is the total number of inliers:

$$S_A = |\{p \in P | e_{p|A} < t\}|$$

When the selection process is finished RANSAC will have generated a set of models, $\{A_1, ..., A_n\}$ and consensus scores $\{S_{A_1}, ..., S_{A_n}\}$. RANSAC then finds the model:

$$A_{\text{max}} = A | \forall A' \in \mathbf{A} : S_{A'} < S_A$$

A_{max} is the model with the highest consensus score $S_{A_{\text{max}}}$. If $S_{A_{\text{max}}} < w|P|$ then A_{max} cannot be the true model since it doesn't describe all the inliers in the scene. In this case the only remaining points must be noise and the recursive process terminates. If $S_{A_{\text{max}}} >= w|P|$ then a transformation describing the motion of an independently moving object has been found. A segment S containing the points $\{p | e_{p|A} < t\}$ is created and these points are removed from the input set. Provided $|P| - |S| > k$ the algorithm recurses on the remaining points to try and find more segments.

When the algorithm works correctly each stage of the procedure should find the largest remaining independently moving object in the scene. Each resulting segment should contain points from a different object and the algorithm will have segmented the input by object.

3.2.3 Normalized projection error

In addition to the standard measure of projection error, a normalized projection error was used. This normalized error is based on a study of the ground truth data given in Chapter 4. The normalized error is based on $\hat{v}$ the distance moved by a tracked point:

$$\hat{v} = \frac{\sum |p_d - p_s|}{|R|}$$

Given $\hat{v}$ the normalized error for a transformation A is given as:

$$\hat{e}_{p|A} = \frac{|Ap_s - p_d|}{\hat{v} + 1}$$

For a magnitude model representing motions with magnitude d the normalized error is:

$$\hat{e}_{p|A} = \frac{|d - |p_d - p_s||}{\hat{v} + 1}$$

Chapter 4

Experiments

4.1 Overview

To evaluate the applicability of motion segmentation towards the domain of feature grouping we used our system to create feature associations. Each association was then evaluated against ground truth. This gave us a general overview as to how well motion segmentation performs with standard methods under non-ideal conditions. We describe the parameters involved in the process and where applicable, we examine the effects those parameters have on performance.

4.2 Data

Our dataset was created in order to accurately reflect real life scenarios where motion segmentation might be applicable. The data was comprised of eight image sequences from three different sources. We chose to work with finished, production quality films to reduce noise effects such as motion blur, focal blur, etc. as much as possible. In addition, every sequence used in our experiment was devoid of special effects and scene cuts. The three source films were Much Ado About Nothing[Bra93] (ado), Valkaama [Bau09] (val), and Route 66 (r66) [Klu04]. The films varied in image size, frame rate, and production quality.

Name	Source	# Frames	Size	# IPs	Camera Motion	# Objects
None_1	Route66	33	400x300	149	none	1
None_2	Valkaama	23	720x405	279.5	none	2
None_3	Valkaama	48	720x405	242.333	none	3
Pan_1	MuchAdo	119	720x480	341.333	pan	1
Pan_2	Valkaama	18	720x405	360.474	pan	2
Pan_3	MuchAdo	73	720x480	280.417	pan	3
Zoom_1	Route66	10	400x300	160.182	zoom	1
Zoom_3	MuchAdo	23	720x480	339.542	zoom	3

Table 4.1: A breakdown of the characteristics of individual scenes. Each scene has been given a unique name based on the last two columns. These last two columns list the underlying camera motion and the number of objects moving independently of the camera. The video which the scene came from is reported in the scene column. Finally, the number of frames in each scene is given here as well as the average number of interest points detected in each frame.

We chose scenes that represented some of the most common motion scenarios that a human or robot might encounter. Table 4.1 describes each of the scenes we chose. The scenes feature three types of camera motion. No camera motion (standing still), panning motion (looking around), and zooming motion (walking forward). The scenes also contained between one and three independently moving objects. The scenes were between 22 and 242 frames long and the average number of interest points per frame varied from 149 to 360.

Figure 4.1 lists a sample labeled image for each of the scenes. Each image was selected from partway into the scene so that tracked points would be available. The arrows represent these tracked interest points. The arrow head falls on the position of the interest point in the current image while the arrow tail originates from the position of the interest point in the previous image. The color of the arrow indicates the ground truth label given to that interest point. The background is always red and each object moving independently of the background is given a unique color.

By looking at figure 4.1 it is possible to get an understanding of the motion occurring in the scene. The train in none_3 and the background in zoom_1 are perhaps the easiest to understand. There are several large motions in pan_1, none_1, none_3, and zoom_1 which are obviously noise. These labels are based on output from the tracker and interest point detector. These systems had difficulty with solid patches of dark color present in these images. It should also be noted that the woman in pan_2 remained mostly still throughout the scene and was thus labeled as background.

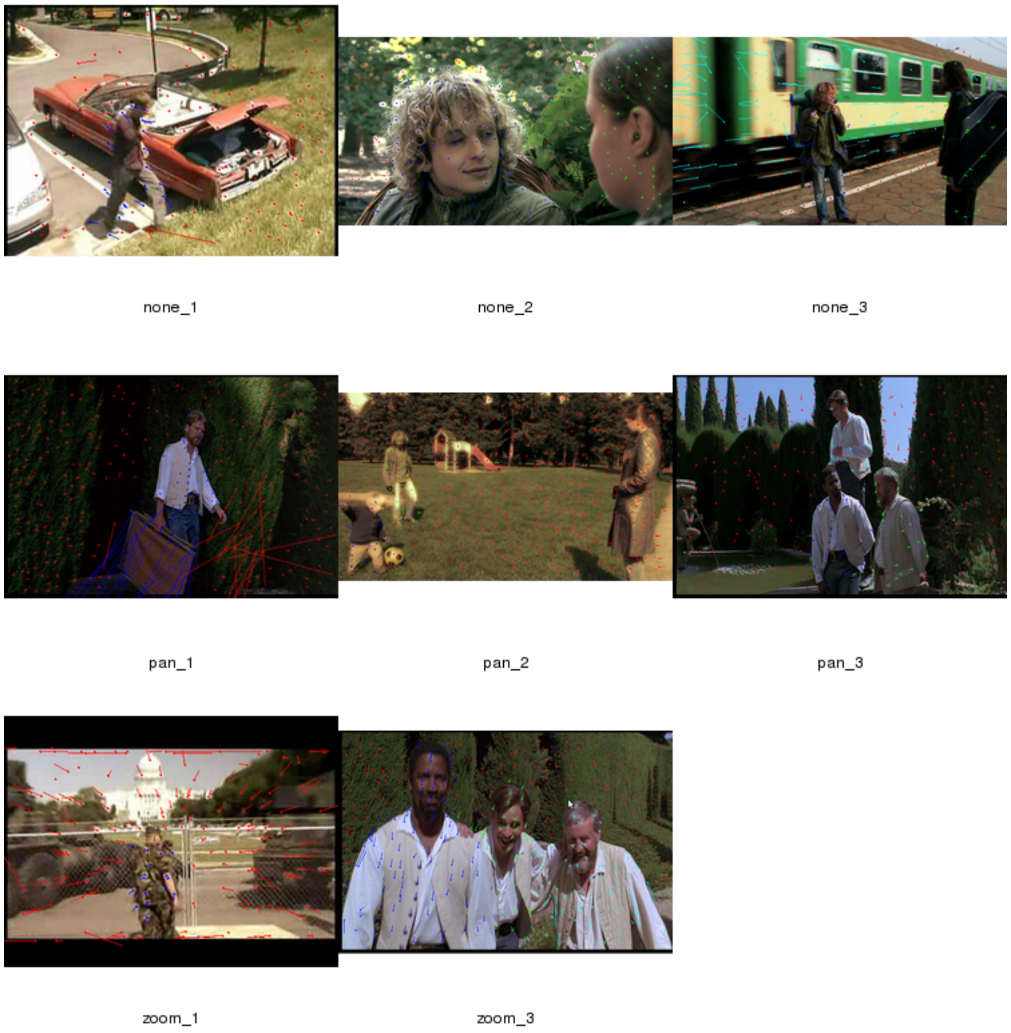

Figure 4.1: Sample images for each of the scenes. The arrows represent tracked interest points. The head of the arrow represents the location of the interest point in the image given. The tail of the arrow represents the position of the interest point in the image immediately preceding the given image. The color of the arrow indicates the ground truth label of the interest point. There should be one color for each object moving independently of the background. The background itself is represented by the color red.

Throughout all the scenes there were a total of 16 independently moving objects and 8 background objects. Of the 16 independently moving objects, 12 of them were adult humans, 2 were human faces, 1 was a human child, and 1 was a train. The objects varied in average size, magnitude of motion and rigidity of motion.

All of the scenes were hand labeled with ground truth data. First, features were extracted from every frame by the previously described salience module of the SeeAsYou system. These features were tracked between frames by the tracking module. The resulting feature correspondences were assigned to either the background or one of the independently moving objects by hand labeling. Point correspondences which appeared to be erroneously tracked were left in place in order to test the performance of the system under noisy conditions. The number of interest points detected for each object depended on the objects size and interest to the salience system.

Detailed descriptions of each of the objects is given in table 4.2. The table gives a name for each object based on the name of the scene the object was present in as well as the label assigned to that object. Labels are represented by colors and the label for each object is also given in the table. These colors match the color of the object in the image given in figure 4.1. The average number of interest points detected on the object per frame as well as the average distance moved by the object per frame is given. Finally, the object is classified into one of five classes.

4.2.1 Data Fitness

In order to predict system performance we also determined the ground truth motion complexity and noise for our data. To determine the degree to which an object's motion could be explained by a simple motion model a selection process was used to find the best ground truth affine transformation for the object at each frame. We then scored the object based on the accuracy of this transformation. Objects with a low score had motion that could be accurately predicted by an affine transformation. Objects with a high score either underwent a complex non-affine motion, were poorly tracked between frames, or failed to be reliably detected by the feature detection system.

Table 4.2: Breakdown of individual objects. This table lists the interesting parameters for each object. The name of the object is based on the scene the object originated from and the label given to the object. More detail on the scene the object originated from can be found in table 4.1. The label describes a color assigned to the object. Colors are unique for a given scene. The color red is always given to background objects. This table also lists the average number of interest points detected per frame as well as the average distance the object moved per frame. Finally the objects are grouped into one of five classes.

Object Name	Scene	# IPs	Motion Distance	Label	Type
N1B	None_1	17.75	6.77	Blue	Adult Human
N1R	None_1	131.25	0.65	Red	Background
N2B	None_2	110.75	1.87	Blue	Adult Face
N2G	None_2	76.375	5.38	Green	Adult Face
N2R	None_2	92.375	0.21	Red	Background
N3B	None_3	40.625	5.51	Blue	Adult Human
N3C	None_3	115.417	27.43	Cyan	Train
N3G	None_3	35.25	5.20	Green	Adult Human
N3R	None_3	51.0417	1.23	Red	Background
P1B	Pan_1	61.6667	22.68	Blue	Adult Human
P1R	Pan_1	279.667	7.00	Red	Background
P2B	Pan_2	8.10526	9.58	Blue	Child Human
P2G	Pan_2	18.6316	6.92	Green	Adult Human
P2R	Pan_2	333.737	6.31	Red	Background
P3B	Pan_3	29.875	2.57	Blue	Adult Human
P3C	Pan_3	22	1.80	Cyan	Adult Human
P3G	Pan_3	25.3333	2.26	Green	Adult Human
P3R	Pan_3	203.208	1.63	Red	Background
Z1B	Zoom_1	26	5.01	Blue	Adult Human
Z1R	Zoom_1	134.182	13.01	Red	Background
Z3B	Zoom_3	95.25	11.98	Blue	Adult Human
Z3C	Zoom_3	59.6667	10.46	Cyan	Adult Human
Z3G	Zoom_3	38.7917	8.96	Green	Adult Human
Z3R	Zoom_3	145.833	5.09	Red	Background

To select a ground truth transformation for an object we evaluated a large number of possible affine transformations seeded by ground truth points on the object. We extracted a set of 50 tracked points from the object to evaluate. If the object had less than 50 tracked points then we used all of the available tracked points. From this set we looked at every possible combination of 3 points. Each combination was analyzed to yield a single affine transformation, A.

To score a transformation we measured the accuracy with which the transformation predicted the underlying motion by measuring the projection error of each point under A. Let p be a tracked point which moved from $p_s = (x_s, y_s)$ to $p_d = (x_d, y_d)$. The projection error for p under A, $e_{p|A}$, can then be calculated by $e_{p|A} = |Ap_s - p_d|$. The projection error for the transformation e_A is then simply the average $e_{p|A}$ across all tracked points in the ground truth set. The transformation with the smallest projection error, $\hat{A}$, was chosen as the most accurate representation of the underlying motion and the projection error, $e_{\hat{A}}$ is the minimal projection error for an object in the given frame.

The average minimal projection error for an object across all frames will be referred to as the fitness score for that object and is a representation of how well an affine motion model can fit the raw point correspondences. The identity transformation, $I|Ix = x$, which represents no motion will always yield a projection error equal to the average distance moved. This means that an upper bound on the projection error is equal to the average distance the object moved.

The fitness score for all objects is given in table 4.3. In this table the average distance moved by the object per frame is given. As well, the fitness score, $e_{\hat{A}}$, is averaged across all frames and listed for each object. The normalized fitness score represents an alternative scoring measure discussed in more detail below. From this table it can be seen that the fitness score appears to be related to the distance the object moved.

The relationship between fitness score and motion is shown in more detail in figure 4.2. The horizontal axis on this figure represents the average distance per frame an object

Object Name	Motion Distance	Fitness Score	Normalized Fitness Score
N1B	6.77	4.13	0.49
N1R	0.65	0.45	0.25
N2B	1.87	1.41	0.40
N2G	5.38	2.98	0.45
N2R	0.22	0.21	0.17
N3B	5.51	4.82	0.63
N3C	27.44	17.83	0.63
N3G	5.21	4.01	0.60
N3R	1.24	1.08	0.36
P1B	22.68	21.46	0.62
P1R	7.01	4.08	0.36
P2B	9.58	2.27	0.02
P2G	6.93	2.17	0.26
P2R	6.32	1.96	0.19
P3B	2.58	1.19	0.32
P3C	1.80	0.66	0.24
P3G	2.26	0.88	0.26
P3R	1.63	0.65	0.26
Z1B	5.02	1.54	0.24
Z1R	13.02	8.02	0.52
Z3B	11.99	5.71	0.45
Z3C	10.46	3.96	0.36
Z3G	8.96	2.73	0.30
Z3R	5.09	1.96	0.26

Table 4.3: Projection error for individual objects. This table shows the projection error, $e_{\hat{A}}$, for each object averaged across all frames. In addition we list the average distance moved per frame for each object similar to Table 4.2. The final column gives the normalized projection error, $\hat{e}_{\hat{A}}$, for each object averaged across all frames. These error measures are described in more detail in the surrounding text.

moved while the vertical access represents the fitness score, $e_{\hat{A}}$, for that object averaged across all frames. A regression line was fitted based on the model that the fitness score could be linearly calculated from the average motion. This line is also given in figure 4.2. This graph appears to show a strong linear relationship between the two properties. Objects which didn't move much scored well while objects that moved a large distance scored poorly. In order to measure this relationship, the correlation between distance moved and the projection error was calculated and it was found to be 0.9275.

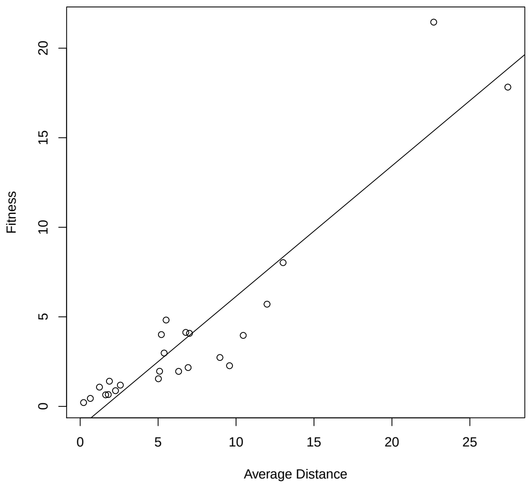

Figure 4.2: Fitness vs. average distance. This figure shows the linear relationship between projection error and average distance. The x axis represents the average distance per frame an object moved. The vertical axis represents the projection error, $e_{\hat{A}}$, averaged across all frames. Each of the points represents one of the objects. In addition, a regression line predicting projection error from motion is laid on top of the plot. The slope of the regression line is 0.73.

The slope of the regression line shown in figure 4.2 is 0.73. Were this slope 1 it would suggest that the ground truth objects were completely impossible to model with an affine transformation since the projection error was equal to the upper bound. A possible reason for such a strong finding is that the tracker seems to be less accurate at larger distances. In order to account for this correlation we also calculated a normalized projection error. Given the average distance an object moved, v, we calculated the normalized projection error for a transformation A and a point p as $\hat{e}_{p|A} = \frac{|Ap_s - p_d|}{v+1}$. The normalized projection error has a lower bound of 0 and an upper bound of 1.

Figure 4.3 lists the normalized fitness error for each of the objects. The distribution of normalized fitness errors are given as box and whisker plots. Each plot is labeled on the horizontal axis with the object represented by the plot. The vertical axis represents the normalized fitness error which have already stated ranges between 0 and 1. The plots are color coded based on the class of the object. The figure shows that there is considerable variation between the objects. Furthermore, a significant number of objects fall into the 0 to 0.6 range which is far enough below 1 that we can assume the objects can be accurately represented by affine motion models. There doesn't appear to be any pattern based on the class of the object.

The inter-class variation is studied in more detail in figure 4.4. This figure shows the average boxplot for each of the object classes similar to figure 4.3. The human child, train, and human face have been grouped into a class labeled other for simplicity. Surprisingly, there is little variation between the classes. It would make sense for the background to perform significantly better since the background is more rigid than the other objects but this doesn't seem to be the case. A two sided t-test was run and it was found that the background and adult human distribution are different $(p = .057)$ however the difference appears to be quite small.

Tests were also run looking for an effect between the number of interest points on an object and performance. Objects that are smaller have fewer points and so it could be possible that it is easier to find a model fitting all the points. However, the correlation was -0.10 suggesting the two are unrelated. A graph of the comparison can be found in the figure 4.5. This horizontal axis shows the average number of interest points detected per frame where the vertical axis shows the normalized fitness score, $\hat{e}_{\hat{A}}$, averaged across all frames. As expected by the low correlation, there appears to be no strong relation between the two.

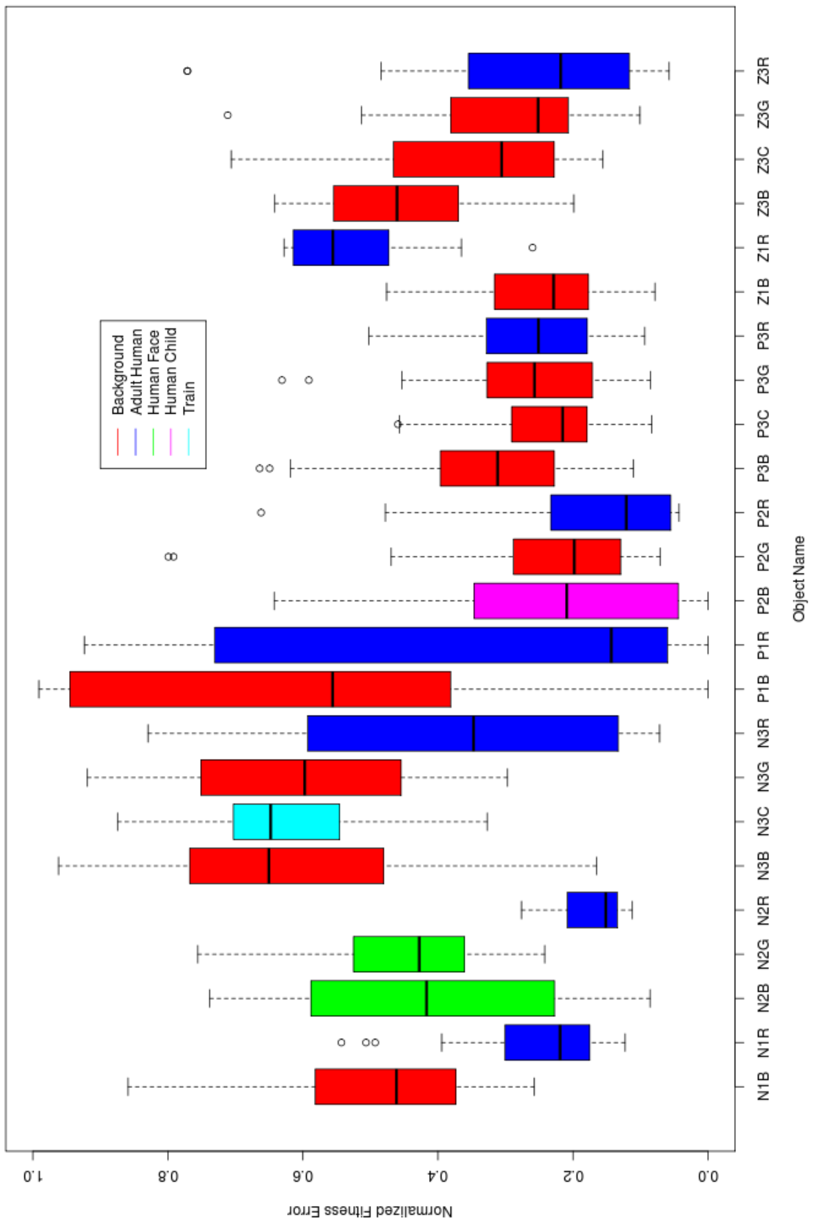

Figure 4.3: Normalized fitness for each object. This figure shows the distribution of the normalized fitness error, $\hat{e}_{\hat{A}}$, for each of the objects. Each object is represented by a box and whisker plot. The object names are listed across the horizontal axis. The vertical axis represents the normalized fitness error. In addition, each plot is color coded based on the class of the object represented.

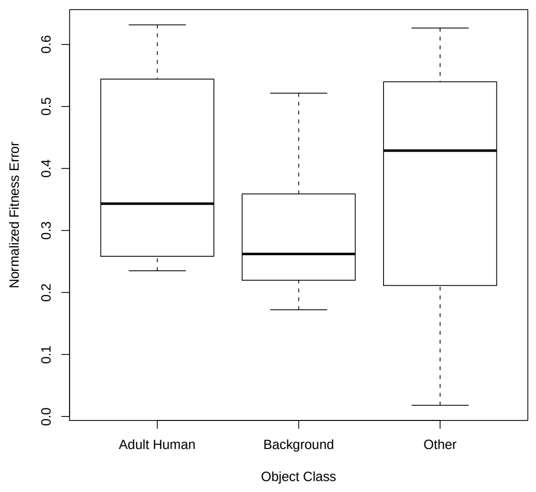

Figure 4.4: Normalized fitness by object type. This figure shows the distribution of the normalized fitness error, $\hat{e}_{\hat{A}}$, for each object class as a box and whisker plot. The horizontal axis lists the object class names. The human face, human child, and train classes have all been grouped into a single other class. The vertical axis represents the normalized fitness error.

4.3 Results

A series of experiments were performed to parameterize the effectiveness of motion segmentation and identify the most important variables in the problem. In the first set of experiments the performance is evaluated as it relates to the eventual problem of forming feature associations. From this one can identify the best motion model and look at possible values for the inlier threshold. Next, the performance of individual objects is measured and that performance is compared with the fitness of the objects. Finally, the normalized error measure is examined and shown to have an effect on threshold setting.

4.3.1 Feature Association

In order to measure the effectiveness of motion segmentation for the task of making feature associations two scoring measures were developed. Given an image I, containing a set of n tracked interest points $P = p_1, ..., p_n$, our system groups the feature points into k segments $\mathbb{S} = (S_1, ...S_n)$ and one noise segment N. Let $C(p) = \text{Label}(p)$ and

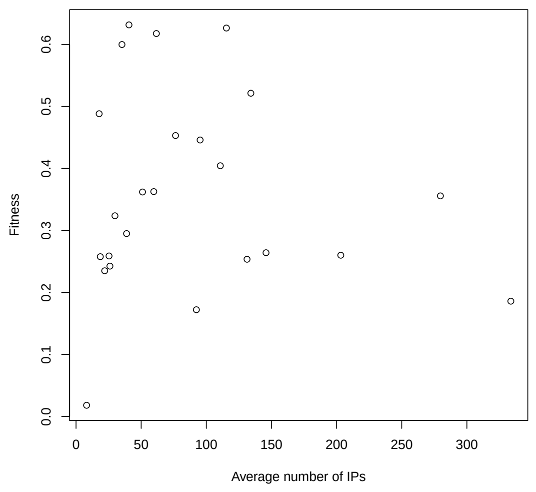

Figure 4.5: Fitness vs. number of IPs. This figure shows the relationship between the projection error and the average number of interest points detected on an object per frame. The horizontal axis gives the average number of features that landed on a given object in each frame. The vertical axis gives the normalized fitness error, $\hat{e}_{\hat{A}}$, discussed in section 4.1 averaged across all frames. There is not a strong linear relationship and so no regression line was fit to the data.

$S(p) = S|S \in \{\mathbb{S}, N\}, p \in S$ be the segment containing p. The first scoring measure is the likelihood that two points have the same label given that they are contained in the same segment:

$$\begin{aligned} L(C_=|S_=) &= \frac{p(C_=|S_=)}{p(C_=)} \\ &= \frac{p(C(p_1) = C(p_2)|S(p_1) = S(p_2)! = N)}{p(C(p_1) = C(p_2))} \end{aligned}$$

If $\mathbb{S}$ is empty then $p(C_=|S_=)$ is defined to be $p(C_=)$ and $L(C_=|S_=) = 1$. This likelihood explains the purity of our segments. If every segment only contain points belonging to a single object then the likelihood will be equal to the maximum, $\frac{1}{p(C_=)}$. The measure favors low inlier thresholds. Low thresholds only segment points which have a high probability of belonging together.

Unfortunately, low thresholds tend to oversegment the image. If there are multiple segments describing a single object we can still score the maximum likelihood, but this is not an ideal segmentation. The oversegmentation issue motivates our second scoring measure. The second scoring measure is the difference, $p(C_=|S_=) - p(C_=|S_{\neq})$. We define $p(C_=|S_{\neq})$ as:

$$p(C_=|S_{\neq}) = p(C(p_1) = C(p_2)|S(p_1) \neq S(p_2))$$

This probability represents the probability that two different segments contain points belonging to the same object. When the threshold is low and the image is oversegmented $p(C_=|S_{\neq})$ will be high and the difference measure will be low. When the segmentation is ideal, $p(C_=|S_{\neq})$ will be 0 and $p(C_=|S_=)$ will be 1 giving us a difference of 1. The difference measure could be thought of as an information gain. When the difference is 0, that means that $S_=$ fails to predict $C_=$ as $C_=$ is just as likely to happen if $S_=$ is not true. When the difference is 1 then points belong in the same label if and only if they are segmented together, which is exactly what we want.

Figure 4.6 shows several sample segmentations for an example image from the pan_3 scene at different inlier thresholds given an affine motion model. Each image is marked with the tracked points that were successfully segmented. Point correspondences that ended up in the noise segment are not shown on the image. The arrows represent the previous and current location of the interest point as they did in Figure 4.1 The color of the arrow represents the segment that the error was placed in. If the colors are the same then the tracked points were placed in the same segment.

From this figure the effect of the inlier threshold can be seen. When the threshold is 0 no motion model is found which explains enough points and all points are placed in the noise segment. As the threshold increases more points are included in actual segments until there are no points left in the noise segment.

An interesting observation is that when the inlier threshold is 1 pixel we have an almost perfect separation of background from foreground, although the foreground is placed in the noise segment. At inlier thresholds of 2 and 3 pixels we have some segmentation of the foreground while maintaining the integrity of the background segment.

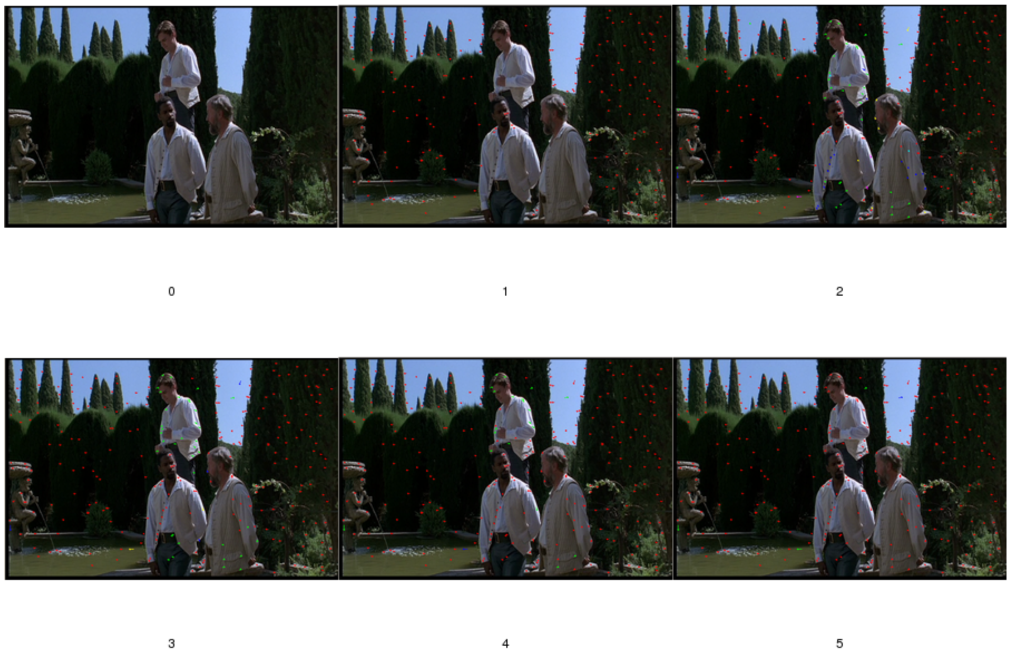

Figure 4.6: Example segmentations at different inlier thresholds. This figure shows the segmentation produced by the system at different inlier thresholds using an affine motion model. All of the images come from the pan_3 scene. Segmented point correspondences are represented as arrows where the head of the arrow is the current position of the point and the tail is the previous position. The color of the arrow represents the segment that the underlying point was placed in. If two points have the same color then they were put in the same segment. Points that were not segmented and classified as noise are not shown.

Once the inlier threshold hits 4 pixels the system begins to severely undersegment. When the inlier threshold is 5 pixels the system placed almost all of the points into a single segment.

Figure 4.7 shows several statistics for the entire pan_3 scene. The left graph shows how the two conditional probabilities described earlier perform. The horizontal axis gives the inlier threshold while the vertical axis gives probabilities from 0 to 1. The flat green line shows the probability that two random points have the same label.

The upper line gives the probability that two points have the same label given they are placed in the same segment. The higher this line is above the flat line the more accurate the segments produced by our system. This line behaves as would be expected. It

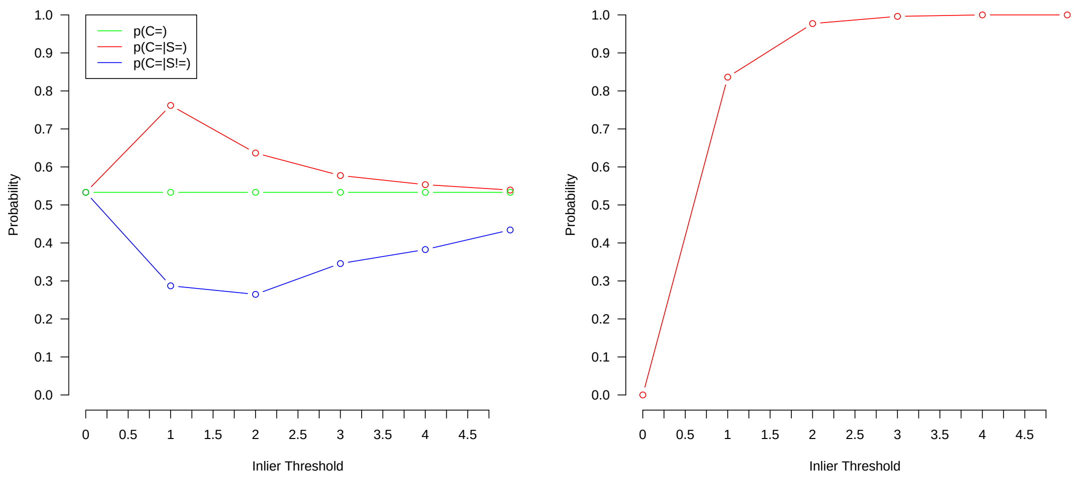

Figure 4.7: Example scores at several inlier thresholds. This figure shows how the scoring measures used in this paper vary with the inlier threshold. The horizontal axis in both graphs shows the inlier threshold. The vertical axis represents probabilities from 0 to 1. The left graph shows the probability that two random points belong in the same label given a)no extra information, b)that the two points were in the same segment, and c)that the two points were in different segments. The right graph shows the percentage of points that were actually segmented and not placed in the noise segment.

peaks at the lowest inlier threshold when the system creates the almost pure background segment and slowly falls back down towards random as the system throws everything into a single segment.

The lower line shows the probability that two points have the same label given they are put in different segments. The higher this line the more likely there are multiple segments for the same label. This line should fall initially as the system oversegments less and less. Then, as the threshold increases the line should head back towards random as the only points not placed in the master segment are so wildly different they are probably tracker errors which are assumed to be randomly distributed throughout the labels.

The right graph of Figure 4.7 shows the number of points which were actually segmented and not placed in the noise segment. The horizontal axis represents the same inlier thresholds as before. The vertical axis gives the percentage of points which were not placed in the noise segment. At a threshold of 0 all points are placed in the noise segment so this measure is 0. As the threshold increases more points are actually segmented until we reach the point at 4 pixels where no points are placed in the noise segment.

4.3.2 Motion Models

Given these scoring measures it is now possible to compare each of the motion models. For this comparison the system was tested on the entire dataset. The scores were averaged across every frame and every scene. Figure 4.8 gives the performance of each of the motion models in terms of our two scoring measures. In both graphs the horizontal axis represents the inlier threshold. In the left graph the vertical axis shows the likelihood score, $L(C_=|S_=)$. In the right graph the vertical axis shows the difference score, $p(C_=|S_=) - p(C_=|S_{\neq})$. Each line in the graphs represents one of the motion models described earlier.

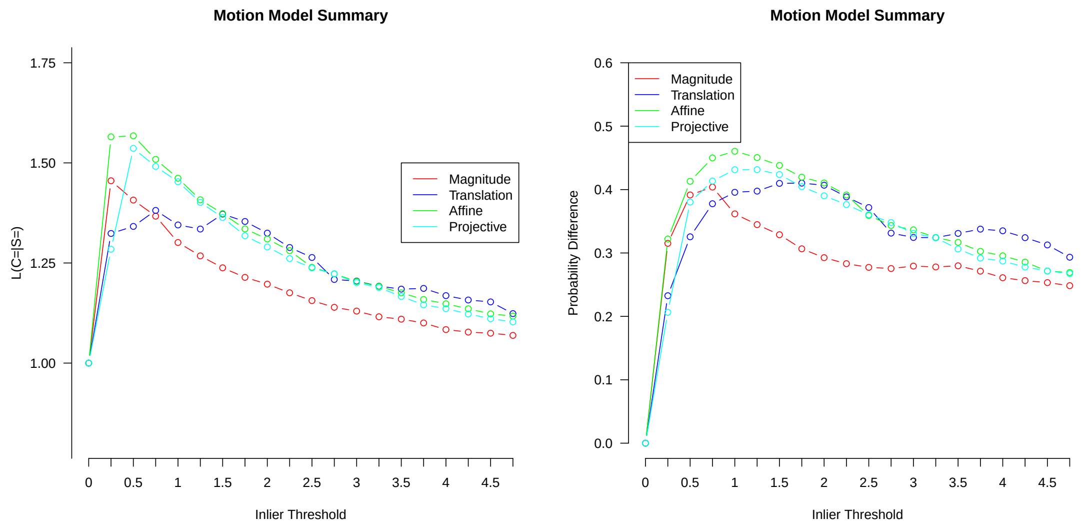

Figure 4.8: Average scores across all scenes in the image. This figure compares the different motion models studied in this paper. The horizontal axis in both graphs represents the inlier threshold in pixels. In the left graph the vertical axis represents the likelihood score. In the right graph the vertical axis represents the difference score. Each line represents a particular motion model. The scores are averaged across every frame of every scene in our dataset.

In both graphs, affine outperforms all other models, including the projective model used by Sivic and Zisserman. However, the difference isn't very large. From the earlier fitness study we saw that most of the objects could be accurately described by an affine motion. This means that the extra variables in the projective model are not needed. It is likely that these extra variables allowed for models which explained more of noise and that is why projective performed slightly worse.

Another point worth noting is that the peaks in the two graphs are different. The likelihood peaks at about a half a pixel where the difference peaks between 1 and 1.5 pixels. As discussed earlier this is because the likelihood score ignores the oversegmentation that happens at the lower thresholds. To confirm this, the individual components of the score can be plotted individually, which is shown in figure 4.9. Similar to figure 4.7, figure 4.9 shows the three probabilities we measured and the effects of the inlier threshold. However, where figure 4.7 only showed results for pan_3, figure 4.9 shows results averaged across all scenes. Figure 4.9 only shows results for the affine motion model.

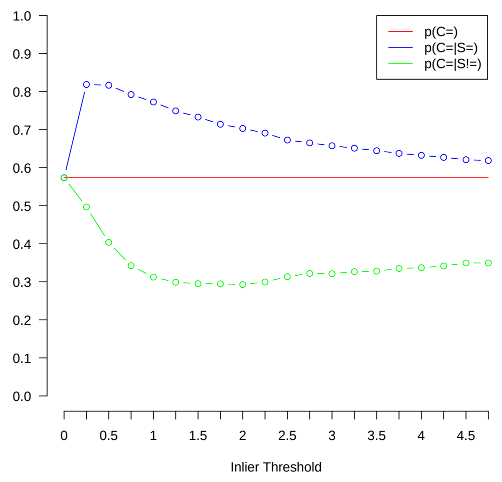

Figure 4.9: Detailed breakdown of the affine model. This graph shows how the probabilities which make up our scores vary with the inlier threshold. The probabilities are for the affine motion model and are averaged across every frame and every scene in our dataset. The horizontal axis represents the inlier threshold in pixels. The vertical graph shows the probability that two random points belong in the same label given a)No extra information, b)That the two points were in the same segment, and c)That the two points were in different segments.

Again, $p(C_=|S_=)$ and $p(C_=|S_{\neq})$ behave exactly as expected. The lower thresholds yield highly accurate segments but oversegment the image. Once the threshold hits 1 pixel the oversegmentation stops. The point where the oversegmentation stops is right at the peak difference score shown in figure 4.8. Beyond 1 pixel the purity of the segments drops but the oversegmentation isn't being reduced.

The fact that the two scores have different thresholds shows that the ideal threshold is task dependent. If the goal is simply to reduce the number of false associations made then a very low threshold is ideal. However, if it is important to associate as much of the object as possible then a slightly higher threshold is better.

4.3.3 Object Performance

Previous experiments have shown the performance of the system overall. In order to understand how performance varies new scores must be developed which give the performance of individual objects. Each object represents a specific label. In an ideal system when points have the same label, x, those points should belong in the same segment. This leads to a likelihood score:

$$\begin{aligned} L(S_=|C_x) &= \frac{p(S_=|C_x)}{p(S_=)} \\ &= \frac{p(S(p_1) = S(p_2) \neq N | C(p_1) = C(p_2) = x)}{p(S(p_1) = S(p_2))} \end{aligned}$$

This score represents the percentage of correct associations found for a given label. When the score is 1 the system has made every possible correct association for the object. Unfortunately, a system could easily get a perfect score by simply segmenting every point into a single segment. However, as before, there is a negative score that can be considered. There are two ways this negative score could be defined. Sticking with a strict conditional probability definition $C_{\neg x}$ is defined as:

$$p(S_=|C_{\neg x}) = p(S(p_1) = S(p_2) | C(p_1) \neq x \vee C(p_2) \neq x)$$

This score ideally would represent the percentage of incorrect associations made. The problem with this score is that points could satisfy the criteria even though they are valid associations. Any two points that both have label $y|y \neq x$ should be in the same segment. If $p(S_=|C_x) - p(S_=|C_{\neq x})$ was used to score an ideal system then there would be a bias towards larger objects. A fairer measure, which only considers points that truly should not be associated, is:

$$p(S_=|C_{\hat{x}}) = p(S(p_1) = S(p_2)|C(p_1) = x \wedge C(p_2) \neq x)$$

This score again represents the percentage of incorrect associations made but requires one side of the association to involve the object we are measuring. To study the effectiveness of this new difference score we again average scores across every frame of every scene. When this score is 1 the system has made every possible incorrect association with this figure which is exactly what would happen if we grouped every point into the same segment. It's possible the system could actually form more incorrect associations than correct associations and the difference score would be negative. Figure 4.10 shows the average $p(S_=|C_x) - p(S_=|C_{\hat{x}})$ for each object.

The horizontal axis of the figure represents the inlier threshold in pixels. The vertical axis represents the difference score. Each line represents a different object in the scene. The color of the line represents the class of the object. This graph shows that background objects outperform all other objects by a considerable amount. This finding reinforces an earlier finding where it was shown in an example image that at low thresholds we could obtain an almost pure representation of the background.

Another effect to notice in figure 4.10 is that the ideal inlier threshold varies between the objects. Intuitively this makes sense since the inlier threshold is in raw pixels and the objects moved at different speeds. The backgrounds all perform well at a very low inlier threshold where the other objects seem to prefer higher inlier thresholds. This is discussed in more detail in a Section 4.3.6.

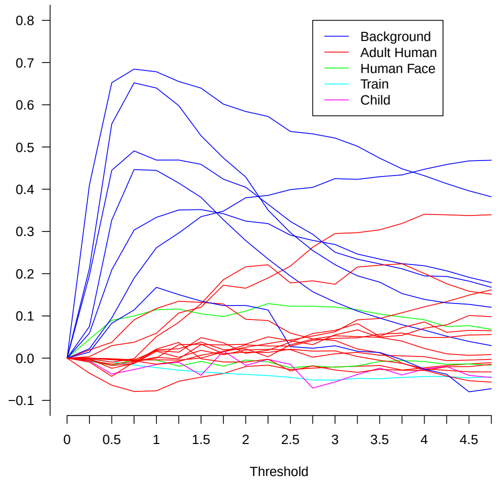

Figure 4.10: Performance by object. This figure shows the performance scores at various inlier thresholds for each object. The horizontal axis represents the inlier threshold. The vertical axis gives the performance score, $p(S_{=}|C_x) - p(S_{=}|C\hat{x})$. Each line represents a different object. The color of the line signifies the class of the object.

4.3.4 Object class performance

Performance appears to be strongly affected by class. To study class variation in more detail the highest possible score for each object is chosen. Figure 4.11 groups max object performance by class. Box and whisker plots are given which show the distribution of performance for each class. The horizontal axis gives the label of the class. Again, the adult face, train, and child classes have been grouped into a single other class. The vertical axis represents the difference score. Background objects outperform all other classes. It is worth noting that the degree to which background objects perform better than foreground objects is significantly higher than it was when we looked at data fitness.

4.3.5 Effect of fitness on performance

It seems reasonable to expect that data fitness should have a large influence on performance. Objects that fit poorly to affine motion models should also segment poorly. To study this the maximum performance for each object was compared to the normalized fitness error for that object. Figure 4.12 shows the actual effect fitness has on perfor-

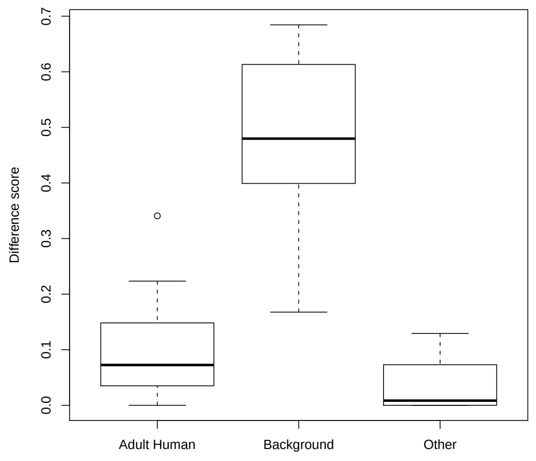

Figure 4.11: Performance by object class. This figure shows the distribution of the maximum performance scores for each object class. The vertical axis represents the performance score, $p(S_{=}|C_x) - p(S_{=}|C\hat{x})$. Box and whisker plots are given for each class. They represent the distribution of the maximum possible performance score for each object in the class. The human face, human child, and train classes are grouped into a class labeled other.

mance. The horizontal axis represents the normalized fitness error for the object. The vertical axis represents the maximum performance for the object. Each point represents an object.

If there is a strong relationship between fitness and performance then we should see the performance fall as the fitness error increases. By looking at the figure it can be seen that this is definitely not the case. The correlation between the two was measured and found to be -0.27. This isn't strong evidence of a relationship (p=.20).

This finding implies that something other than the fitness is influencing performance. There is something special about the background beside its ability to be easily explained by an affine model that is influencing performance. This is discussed further in our conclusions and something we hope to investigate in future work.

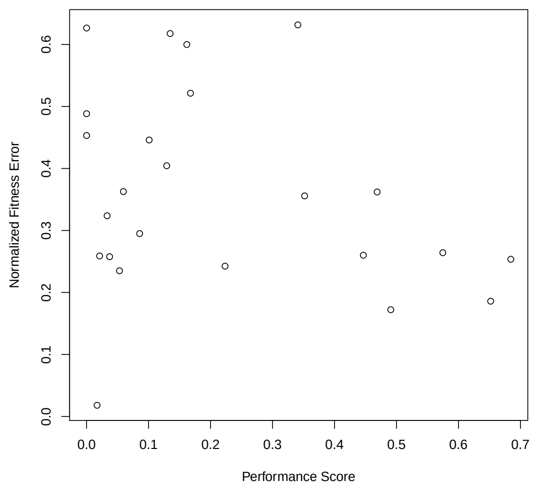

Figure 4.12: Fitness vs. Performance. This figure shows the relationship between normalized fitness error and object performance. The horizontal axis represents the normalized fitness error. The vertical axis represents the maximum possible performance score for the object. Each point represents an object.

4.3.6 Normalized projection error

In addition to the standard projection error the system can calculate a normalized projection error. The most interesting effect caused by the normalized projection error is that it allows for a better selection of inlier threshold. Figure 4.13 demonstrates this effect. Similar to figure 4.10, figure 4.13 shows the performance scores for each object given an inlier threshold. The horizontal axis now goes between 0 and 1 since the threshold is now based on the normalized projection error. It seems that more objects peak around the same point in figure 4.13 than did in figure 4.10.

To measure this effect quantitatively a scoring method was devised which measured the effectiveness of an inlier threshold across all objects. To give each object equal weight the performance scores were normalized. Given inlier threshold $t \in T$ where T is the set of all measured thresholds, let $f(t, x)$ be $p(S_=|C_x) - p(S_=|C_{\hat{x}})$ given object x and inlier threshold t. In other words, f is the difference score for a given threshold and object. Also, let $t_{x,\text{max}} = t|\forall t' \in \{T - t\} : f(t, x) > f(t', x)$ and $t_{x,\text{min}} = t|\forall t' \in \{T - t\} : f(t, x) < f(t', x)$. We can then look at:

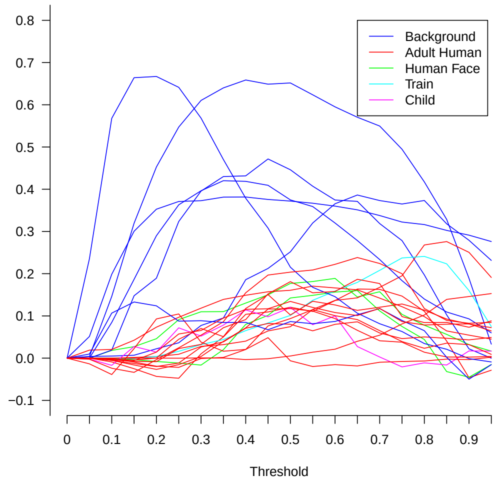

Figure 4.13: Performance by object (normalized error). This figure shows the performance scores at various inlier thresholds for each object when using the normalized projection error. The horizontal axis represents the inlier threshold. The vertical axis gives the performance score, $p(S_=|C_x) - p(S_=|C\hat{x})$. Each line represents a different object. The color of the line signifies the class of the object.

$$g(t,x) = \frac{f(t,x) - t_{x,\text{min}}}{t_{x,\text{max}} - t_{x,\text{min}}}$$

The advantage of $g(t,x)$ is that given a fixed object x' the range of $g(t,x') = [0,1]$. This gives all objects equal weight when looking for an ideal threshold. If we let X be the set of all our labels $\{x_1, ..., x_{24}\}$ we can look at $G(t)$, a measure of how ideal the threshold t is across all objects. $G(t)$ is defined as:

$$G(t) = \sum_{i=1}^{24} g(t, x_i)$$

Figure 4.14 uses $G(t)$ to compare the standard projection error and the normalized projection error for threshold selection. The vertical axis represents $G(t)$, the scoring metric for our threshold. The lower horizontal axis belongs to the blue line and shows the inlier threshold for the standard projection error which ranged from 0 to 5 in this work. The upper horizontal axis belongs to the red line and shows the inlier threshold for the normalized projection error, which ranged from 0 to 1.

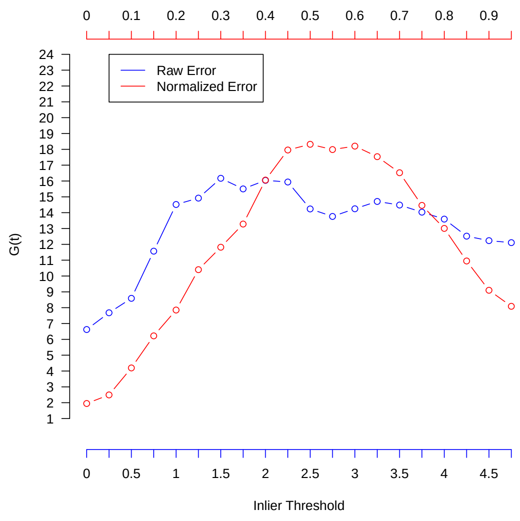

Figure 4.14: Raw error vs. normalized error. This figure shows how threshold selection changes depending on the projection error used. The vertical axis represents the threshold score, $G(t)$, which ranges from 0 to 24. The blue horizontal axis and blue line represents the threshold values and performance for the standard projection error. The red horizontal axis and red line represent the threshold values and performance for the normalized projection error.

It can be seen that the normalized error curve peaks higher than the standard error curve. This shows that by using the normalized projection error it is possible to pick a threshold that fits more objects. In addition the normalized error curve forms a much sharper bell curve than the standard projection error which may not represent a bell curve at all. The standard projection error uses an inlier threshold in pixels and so it tends to be different for each object depending on the distance moved by the object. The normalized projection error threshold is fixed between 0 and 1 for all objects and it's effect is constant across all objects.

Chapter 5

Conclusions

The goal of this research was to evaluate the applicability of motion segmentation towards forming object percepts. The experiments performed have shown that there are feature associations to be made from motion information. These associations can be accurately detected by a system such as the one described in this paper. Most of the associations made will simply be segmenting background from foreground but there are some associations made on foreground objects as well.

Both affine and projective motion models have been shown to perform about the same. In this work the affine motion model performed slightly better. This is likely due to the fact that most of the objects were shown to be well described by an affine model and so a projective model was unnecessary. Both of the models had near identical behavior when it came to a selection of threshold.

The system was shown to oversegment at lower thresholds and undersegment at higher thresholds as expected. It was shown that at the lowest thresholds, segments tended to be very pure. The higher the thresholds the more the segments explained the entire object they were covering but they tended to become less pure. This leads to a tradeoff that systems hoping to use this information must make when it comes to selecting a threshold. One method for resolving the tradeoff was presented with the difference score which gave a slightly higher threshold than the pure likelihood threshold.

In addition we presented an alternative error measure which can lead to an easier selection of ideal threshold. It was demonstrated that the projection error of the best fit model tends to be influenced by the distance the object moved. This means that a threshold which is based on raw pixels will give a different ideal for models moving at different speeds. Normalizing the error by the distance the object moved created a threshold that is independent of the motion of the object.

Finally this paper has shown the existence of some unmeasured phenomenon that affects the quality of performance. It was shown that the data fitness alone could not account for the variations in performance amongst objects. This implies that an object whose motion is well described by an affine motion model will not necessarily segment well. This has implications for anyone trying to develop a better segmentation system. It isn't guaranteed that simply improving the input by finding a more descriptive transformation, reducing the tracker error, or selecting only rigid objects would have a significant beneficial effect on the system.

5.1 Future Work

In the course of this work several opportunities arose for experiments that were out of the scope of this thesis which could improve this work. The most prominent is to investigate the discrepancy between data fitness and performance. This would entail identifying some property or collection of properties of the ground truth data that explains the performance. One hypothesis we propose is that the highest performing motions are also the most distinct from the surrounding motions.

Another weakness is that all of the motion models studied were transformation based models. It would be interesting to compare performance between models based on transformations with models based on trajectories or flow. It would also be interesting to study the accuracy of algebraic models for calculating the motion directly instead of random sampling through RANSAC.

This paper studied the effect of motion without any prior knowledge. In reality, there are a number of other methods for forming spatial associations. Each of these methods presents a set of associations unique to that particular algorithm. In addition, existing associations between temporal features provide prior knowledge. One could study how motion segmentation could be improved by prior segmentations given from other methods.

The evaluation presented here used a simple measure to examine the quality of the output as a prediction towards performance in a larger system. It would be useful to take the segmentation algorithm used in this work and compare the effectiveness of percept formation systems with and without motion segmentation. One could also examine the percepts that were formed by motion segmentation that would not have been formed otherwise. This would provide a higher level evaluation which may be more accurate for percept based tasks.

REFERENCES

[Bau09] Tim Baumann, editor. *Valkaama*. www.valkaama.com, 2009.

[Bou00] J.Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker : description of the algorithm. Technical report, Intel, Microprocessor Research Labs, 2000.

[Bra93] Kenneth Branagh, editor. *Much Ado About Nothing*. BBC Films, 1993.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *CSUR*, 41(3), 2009.

[DT10] Bruce Draper and Lucy Troup. Seeasyou: Modeling object and scene recognition. *The Journal of Vision*, 2010. Paper under review.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FFFP04] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR*, page 178, 2004.

[FZ00] Andrew W. Fitzgibbon and Andrew Zisserman. Multibody structure and motion: 3-d reconstruction of independently moving objects. *LNCS*, 1842:891–906, 2000.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *The Fourth Alvey Vision Conference*, 1988.

[Klu04] Stefan Kluge, editor. *Route 66*. route66.vebfilm.net, 2004.

[LGF10] Dahua Lin, Eric Grimson, and John Fisher. Modeling and estimating persistent motion with geometric flows. In *CVPR*, 2010.

[LK81] Bruce Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *DARPA IU Workshop*, pages 121 – 130, 1981.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[Luc84] Bruce Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Carnegie-Mellon University, 1984.

[MTS$^+$05] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65(1–2):43–72, 2005.

[RA80] J. W. Roach and J. K. Aggarwal. Determining the movement of objects from a sequence of images. *PAMI*, 2:554–562, 1980.

[RLSP07] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Segmenting, modeling, and matching video clips containing multiple moving objects. *PAMI*, 29(3):477–491, 2007.

[SMB00] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *IJCV*, 37(2):151–172, 2000.

[SSZ05] Josef Sivic, Frederik Schaffalitzky, and Andrew Zisserman. Object level grouping for video shots. *IJCV*, 67(2):189–210, 2005.

[SWRC06] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. *LNCS*, 3951:1–15, 2006.

[SZ03] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, volume 2, page 1470, 2003.

[TK92] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *IJCV*, 9(2):137–154, 1992.

[Tor95] Philip H. S. Torr. *Motion Segmentation and Outlier Detection*. PhD thesis, University of Oxford, 1995.

[TZ00] P. H. S. Torr and A. Zisserman. Feature based methods for structure and motion estimation. *LNCS*, 1883:278–294, 2000.

[Ull79] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society Biological Sciences*, 203(1153):405–426, 1979.

[VMSS05] Ren Vidal, Yi Ma, Stefano Soatto, and Shankar Sastry. Two-view multibody structure from motion. *IJCV*, 68(1):7–25, 2005.

[YJS06] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *CSUR*, 38(4), 2006.