MOTION TRACKING USING FEATURE POINT CLUSTERS

by

ROBERT L. FOSTER JR.

B.S., Virginia Tech, 2004

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF COMPUTING AND INFORMATION SCIENCES
COLLEGE OF ENGINEERING

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2008

Approved by;

Co-Major Professor
Dr. David A. Gustafson

Approved by;

Co-Major Professor
Dr. William H. Hsu

Copyright

Abstract

In this study, we identify a new method of tracking motion over a sequence of images using feature point clusters. We identify and implement a system that takes as input a sequence of images and generates clusters of SIFT features using the K-Means clustering algorithm. Every time the system processes an image it compares each new cluster to the clusters of previous images, which it stores in a local cache. When at least 25% of the SIFT features that compose a cluster match a cluster in the local cache, the system uses the centroid of both clusters in order to determine the direction of travel. To establish a direction of travel, we calculate the slope of the line connecting the centroid of two clusters relative to their Cartesian coordinates in the secondary image. In an experiment using a P3-AT mobile robotic agent equipped with a digital camera, the system receives and processes a sequence of eight images. Experimental results show that the system is able to identify and track the motion of objects using SIFT feature clusters more efficiently when applying spatial outlier detection prior to generating clusters.

Table of Contents

List of Figures

List of Tables

CHAPTER 1 - Introduction

Tracking the motion of objects is a key task in computer vision. Through computer vision we analyze information relating to motion in a sequence of images or video. Robot vision is a subtask of computer vision in which robots collect information from cameras or other optical sensors in order to detect or identify objects in their surroundings. Information that a robot retrieves from its sensors provide the information about the location and possibly the size of an object or obstacle. In some tasks it is essential that a robot be capable of determining if objects are stationary or in motion in order to complete the task. Robots use knowledge pertaining to the motion of an object such as the direction of travel for path planning, obstacle avoidance, and identifying traffic patterns.

In this thesis we hypothesize that it is possible to identify and track the motion of objects in 2D images using clusters of Scale Invariant Feature Transforms keys, from this point we refer to them as to as SIFT features [3]. First, we identify some of the existing problems in object identification and motion tracking. Then, we identify a method for recognizing and extracting SIFT features from an image. Next we identify a method for generating clusters of SIFT features to represent objects in our 2D image. Once we generate clusters of SIFT features to represent objects, we identify object matches over a sequence of images. Finally, for each new cluster that we match to a cluster from a prior image in the sequence, we identify its direction of travel.

To identify and extract SIFT features we use an implementation of David Lowe's SIFT feature identification algorithm [3]. Because one of our objectives is to identify objects from 2D images, it is necessary to define properties of objects within an image [3]. Problems that arise when identifying objects in 2D images include: objects in motion, physical appearance, unusual poses, lighting, reflections, and loss of 3D projection [4]. In Lowe's study, "Object recognition From Local Scale Invariant Features" [7], it is proven that SIFT features are strong enough to identify objects from 2D images even if we manipulate the image. We rely on this fact in assuming that the use of SIFT features minimize the problems above. In a chapter two we

examine SIFT in further detail.

To generate clusters of SIFT features, we must identify a clustering algorithm appropriate to the task we would like to accomplish. In "A Survey of Clustering Algorithms" [6], Rui Xu establishes a basis for clustering on a set of four common principals:

1. Feature Selection or Extraction
2. Clustering Algorithm Design or Selection
3. Cluster Validation
4. Results Interpretation

Each principal identifies a key step in generating data clusters. We explore these steps further in chapter two. We select SIFT features as the type of data to cluster and identify a method for extracting these features using David Lowe's algorithm. The K-Means clustering algorithm is a standard partitioning algorithm frequently used in computer vision. Because our goal is to track the motion of objects over a sequence of images, speed in generating clusters is a priority. In that aspect it is important to note that generally K-Means completes in a number of iterations that is less than the number of points in the data set. In this thesis we identify the K-Means algorithm as our method for clustering. In chapter three we discuss the design details of the basic K-Means algorithm and the modifications necessary to generate clusters of SIFT features. Occasionally, we refer to SIFT feature clusters simply as objects.

To identify object matches among clusters of SIFT features (objects), we implement a matching scheme to compare each individual SIFT feature in a cluster to those of previously identified clusters. Clusters identified in an image are stored in a cache. For each cached object we identify the number of points corresponding to the new object. If the number of points that match is at least 25% of the points in the new image we identify it as a match. We chose 25% as a minimal match rate relative to the set of boundary conditions for our experiment. The basis for these boundaries depend upon the initialization of the SIFT algorithm which determines the number of SIFT features to be created for an image, and the number of clusters that the clustering algorithm generates. Using this information we empirically determine that 25% is an acceptable match rate to identify a portion of an object by observing the number of SIFT features in an object. We consider this the most reasonable match unless we identify multiple matching clusters in the cache. If there are multiple matches we perform further analysis to determine

which the best fit is. We further discuss determining best-fit matches, caching objects, and obstructed objects in chapter four.

Motion tracking of matching SIFT feature clusters is a post processing step in our system in which we identify a direction of travel for each new cluster match. We define a minimum set of directions of travel including North, East, South, West, Northwest, Northeast, Southeast, and Southwest. The first time we identify a new cluster or encounter a cluster without a match we consider it stationary. The direction of travel is computed as a function of the slope of the line connecting the mean point of a cluster in the local cache to the mean point of the new cluster. More precisely, we compute the slope of the line connecting the centroid of two clusters by dividing the original image into four quadrants relative to the x and y-axis. If the orientation of the direction changes between images, then it is necessary to offset the x and y-axis. We cannot guarantee that two objects that match on all points are exactly the same due to problems such as duplicates and reflections which may introduce uncertainty. These invariants require us to define a measure to evaluate the predicted direction of travel over all object matches in a sequence of images. We discuss this measure in chapter five as we identify measurement and metrics to evaluate the entire system.

CHAPTER 2 - Background & Other Approaches

In this chapter, we establish a background in research relating to the topics we cover in order to track motion using feature point clusters. We collect background information in four different topics including scale invariant feature transform (SIFT), clustering algorithms, identifying 3D objects from 2D images, and inferring 3D people from 2D images. For each topic, we reference the relevant pieces of literature and describe the authors' important contributions to this research. Additionally, some of the topics touch on alternative approaches to solving problems relating to the topic. Some of the topics, such as obstructions that impair object recognition are reoccurring themes that heavily influence the system design of the experiment in this research.

Scale Invariant Feature Transform (SIFT)

David Lowe's "Object Recognition from Local Scale-Invariant Features" [7], introduces the idea of using a new class of local image features to identify 3D objects from 2D images. Lowe sets several requirements of these features to ensure reliability and consistency among features. These requirements include invariance to image scaling, translation, and partial invariance to illumination changes and affine (3D projection). The algorithm begins by identifying the local maxima or minima of a difference of Gaussian function. Each point is used to generate a feature vector, then Lowe derives SIFT features from images by blurring gradient locations. Figure 1 (extracted from "Object Recognition from Local Scale-Invariant Features" [7]) shows three objects for which Lowe identifies SIFT features, the second image shows a new arrangement of the objects overlapping in a chair, and in third image smaller rectangles are drawn to represent the SIFT features that Lowe recognizes after rearranging the objects (note that the larger rectangles around the borders of the objects are not SIFT features). It is important to note that Lowe's implementation of SIFT ignores color content and thus operates on grayscale images.

Figure 1: A SIFT Example [7]

In addition to the vector of descriptors, SIFT features maintain scale, orientation, and location for each point. Lowe's test results, shown in Table 1: SIFT Performance [7], are given in percentage matches of feature points. In the test, Lowe first identifies SIFT features in an original image, and predicts the location of those features in the image after performing transformations. Lowe performs various transformations to images including: changes in contrast, intensity, rotation, scale, and stretching. Then, he compares the features from the original image to the features in the image after transformation. Each value in the table is a

percentage of how closely features in the original image match to features in the transformed image by location, scale, and orientation. The results over a 20 image sample for the combined transformations yield a 78.6% match rate on location and scale, and 71.8% over orientation. It is shown that stretching images provides the lowest match percentage of location and scale with a 77.7% match on average, and 65.0% match rate on average over orientation. It should also be noted that pixel noise still allows a 90.3% match rate in location and scale as well as 88.4% in orientation.

Table 1: SIFT Performance [7]

Image transformation	Match %	Ori %
A. Increase contrast by 1.2	89.0	86.6
B. Decrease intensity by 0.2	88.5	85.9
C. Rotate by 20 degrees	85.4	81.0
D. Scale by 0.7	85.1	80.3
E. Stretch by 1.2	83.5	76.1
F. Stretch by 1.5	77.7	65.0
G. Add 10% pixel noise	90.3	88.4
H. All of A,B,C,D,E,G.	78.6	71.8

To obtain local image descriptors, Lowe performs several operations on the pixels of the given image. Using two spatial dimensions and linear interpolation on each pixel in the image and its eight nearest neighbors, Lowe introduces a blurring affect. He explains that finding an exact SIFT feature match over multiple SIFT feature sets has a high complexity so he uses the best-bin-first search method to identify nearest neighbors. In addition he claims that using a Hough transform to search for matching keys is an efficient way to cluster reliable model hypotheses. Identifying nearest neighbors and clustering are also tasks in this study.

A Survey of Clustering Algorithms

Xu describes feature selection as choosing distinguishing features from a set of candidates [6]. Because our goal is to classify data, the data must have qualities that allow us to categorize it. We refer to the collection of similar qualities that a group of data shares as

features. Note that these are not the same as the SIFT feature that Lowe identifies, but generic features that are more like attributes of a data set. Feature identification and selection are the first steps towards clustering data. Xu states that ideal features are those that are useful in distinguishing patterns, immune to noise, and easy extracted and interpreted.

Features should be strong enough so that when generating clusters, it is possible to classify each feature into one of the existing clusters. If a feature is not classifiable we can identify it as an outlier of the given data set. This type of feature provides noise to the clustering method so we must address such features on an individual basis or make modifications to feature specification so that clustering is immune to noise. We can use spatial outlier detection to re-specify or strengthen features.

Clustering Algorithm Design or Selection

Upon identifying common features of a data set, we may begin to design or select a clustering algorithm. In this step it is important to identify a relationship between features. Normally we base the relationship on proximity, which requires us to generate a distance function to map features. After identifying the relationship between features and choosing a measure of proximity we must establish criterion for clustering. Xu states this as an optimization problem with several existing solutions. In many cases this step involves partitioning and can also introduce randomness. Selecting or designing the algorithm necessary to group similar features is heavily dependent on the types of data, precision requirements, and available computing resources.

Cluster Validation

An essential step in the process of designing or selecting an algorithm is the validation of clusters. We must ensure that the algorithm generates clusters only when valid data is present, and then we must attempt to verify the quality of the clusters that the algorithm generates. Xu defines three test criteria based on the clustering structures including external indices, internal indices, and relative indices with respect to 3 different clustering models respectively including partitioning, hierarchical clustering, and individual clustering. Essentially, we must define control data to test against as well as a benchmark for acceptable cluster identification.

It is possible to compare resulting clusters to expected results or test criterion. Specifically in the case of partitioning, it is useful to run controlled experiments with data whose

features are well defined. In this case it is possible to validate the features composing the cluster that the algorithm generates. If features are not a part of the cluster that we expect, then we can detect potential errors in the algorithm and representation of features.

Results Interpretation

In this step, we validate the results, or gather what Xu defines "meaningful insight" [6] from the original data. Our goal is to interpret the results in such a way that they can be useful in solving the initial problem. We must further analyze the resulting data in order to verify the reliability of our results. Additionally, we determine whether the cluster information is useful enough to construct new data sets for further analysis. Consider Figure 2: A Clustering Example, which shows the result of clustering a set of data (numbers one through seven). Suppose that in the example we identify the feature to cluster on as the integer value of each data point. When we interpret the results, we discover that clusters that share the same integer value also share the same color. Given the representation of data after clustering, this observation seems obvious. However, prior to clustering the data may be spread out such that the data points overlap, and the only known fact is that they are integers.

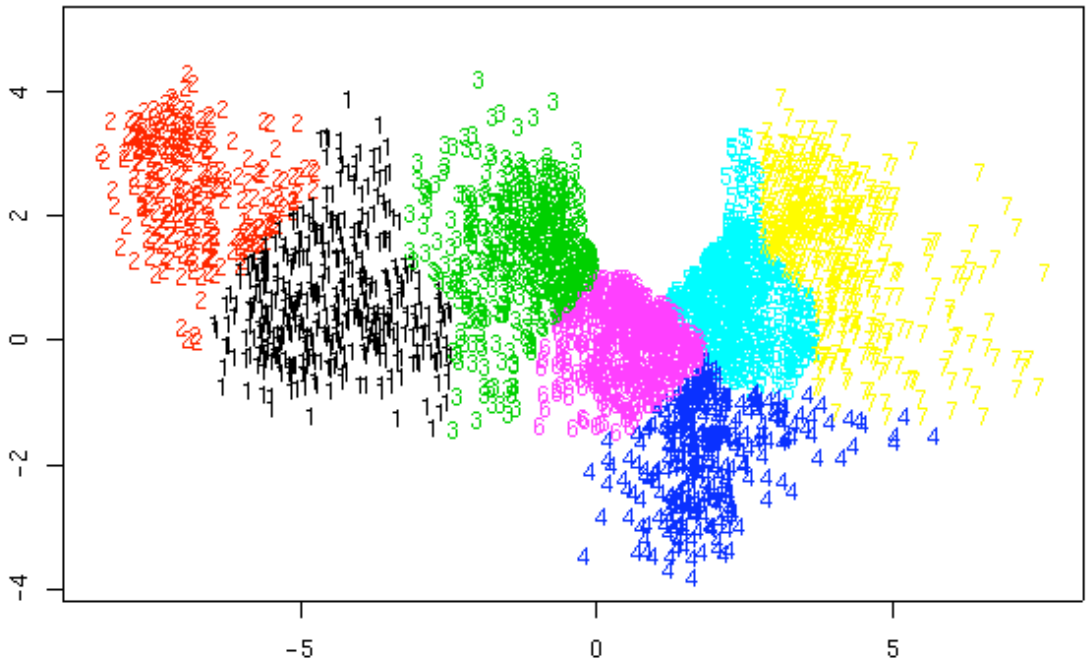

Figure 2: A Clustering Example

Thus a cluster of feature data may provide a new layer of data to be interpreted that could not be expressed in a single feature. Application of the resulting data could present new problems or provide insight into a different approach of solving existing problems.

Object Identification from 2D Images

A previous approach to object identification as described by Mauro Barni in "A Robust Fuzzy Clustering Algorithm for the Classification of Remote Sensing Images" [5] is to generate clusters of pixels. In classical approach to this concept one must know the number of clusters that should be obtained in advance. In this method, a valid cluster is one whose pixel value sums to the value of one. This feature classification pattern fails to meet Xu's feature selection requirement in that it is not immune to noise. In some cases it is possible that we must discard noisy pixels indicating that they do not belong to a particular cluster. This corresponds to a type of feature strengthening as specified by Xu. One could also implement an algorithm that would discard these noisy pixels or outliers or further analyze them to determine which classification set they should belong to as a sort of outlier detection.

Barni [5] proposes a new fuzzy clustering algorithm over the pixels of an image in which the feature classification restraint of summing to zero is relaxed and introduces a probabilistic constraint. Because the original model does not handle noisy cases in which a pixel is not a member of any cluster, Barni introduces Alternate Constraint Clustering. This technique ensures that every pixel has membership within some cluster. The algorithm uses the classical probabilistic constraint for good pixels deemed as good (easily classified) and an r-constraint for noisy pixels. This example of feature extraction complies with the standards described by Xu and serves as an example of how to strengthen features. An alternative is to identify such features as outliers from the data set because they do not provide any useful information. In this study we choose not to generate clusters of pixels because they may lack consistency. The lack of consistency may be due to changes in lighting or shadows thus SIFT feature clusters are more appropriate.

Inferring 3D people from 2D images

In Michael J. Black's presentation of "Inferring 3D people from 2D images" [4], he considers the idea of capturing humans in motion through 2D images. Black discusses using of

markerless motion capture, a system that identifies a model to represent humans in various poses. The system tracks motion by identifying changes in poses using the model. Black identifies the difficulty of tracking people as a factor of people's physical appearances. This applies to object recognition and motion tracking because we must consider generic moving obstacles that may also have moving parts. Other difficulty in object tracking through 2D images includes loss of 3D projection, unusual poses, low contrast lighting and reflections, as well as obstructions. Clothing, lighting, and obstruction play a large role in tracking motion between multiple images as it may be difficult to identify a pose or even the same object if there is another object obstructing it or the image misrepresents objects due to lighting reflections or refractions. The image in Figure 3 exhibits several of these problems including unusual poses and obstruction. Additionally similar poses in people with approximately the same dimensions and colored clothing may appear to be the same person in different images.

Figure 3: Unusual pose, large range of movement, and obstructions

Large motions pose a large problem in motion tracking because blurs and other ambiguities can be introduced into an image. In particular when tracking people as objects or other items with multiple moving parts it can be difficult to identify the path of travel of the entire object as opposed to one moving part. Black states 4 requirements in order to track motion of objects with multiple moving parts:

1. Represent uncertainty and multiple hypotheses.
2. Model complex movements.
3. Exploit multiple image cues in a robust fashion.

4. Integrate information over time.

Black also considers generic background statistics in 2D images using a Bayesian formulation. Filtering out background statistics plays an essential role in both precision on motion tracking and speed. With fewer background images one can process a motion-tracking algorithm more quickly while providing more accurate results by eliminating unnecessary data points. The Bayesian model uses the notion of prior and next cues to learn particular motion patterns. These can be learned over a sequence of images of moving objects.

It is possible to track motion in not only poses but also in direction. If we can identify specific objects as a whole we can identify a direction of travel for those objects between 2D images assuming that we are able to overcome the difficulties Black identifies. Even when tracking the location of stationary objects without moving parts between 2D images it is possible that lighting reflections and refractions may distort results making it difficult to identify even "simple" objects. We do not use this approach because modeling requires a significant amount of overhead, and the approach is directed only towards tracking the motion of people.

CHAPTER 3 - Clustering SIFT features & Tracking Motion:

The K-Means algorithm is a partitioning algorithm for clustering and classification of data. The algorithm partitions a data set of n features into k regions based on the proximity of features. The number of regions k must be specified but may be unknown, or guessed. There are several common mathematical functions that estimate a number of clusters for the algorithm to generate. The proximity of features is determined using a distance function given for a particular feature type. K-Means partitions a data set using Lloyds Algorithm in order to find a solution to a K-Means problem [2]. The algorithm uses an iterative heuristic such that once it partitions the initial set of *n* features, it calculates the clusters mean. It then uses the distance function to determine which cluster to add the given feature. The algorithm repeats each step until no features remain to assign [2]. Figure 4 (extracted from Wikipedia [2]) provides an illustration of the K-Means algorithm.

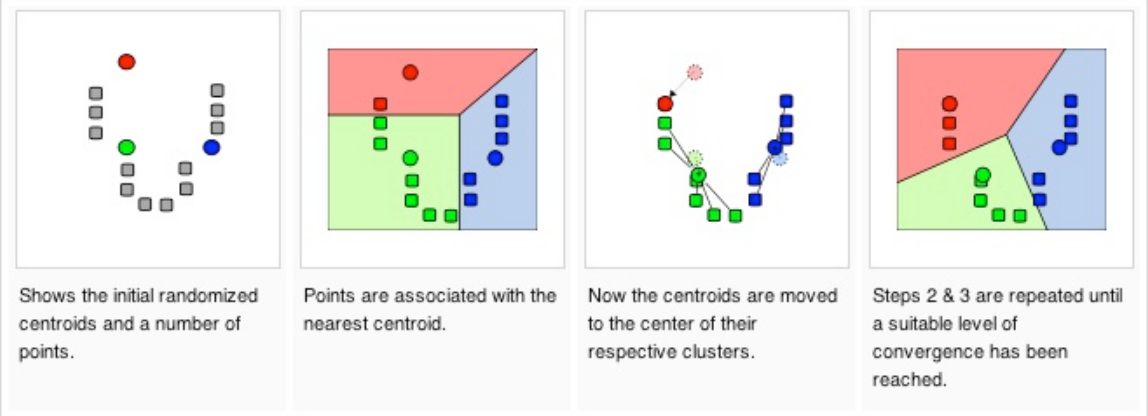

Figure 4: A K-Means illustration [2]

The process of developing the modified K-Means algorithm involves several steps. Here we discuss topics that determine how we design this component of the system, its performance, and the results. First, we give a description of relevant objects and define a function to determine the initial value for *k*. Then, we define the distance function and consider the random nature of K-Means. Next, we discuss cluster validation, clusters as objects, and irrelevant

objects. Finally, we identify two spatial outlier algorithms; consider cluster caching, and everything together to define a method for tracking motion.

Relevant Objects and Choosing an Initial Value for *k*

The process of clustering in order to identify objects requires that we first define which objects we consider relevant. We base this decision on the number of SIFT features in our set of *n* objects. The next task is choosing an initial value for *k*. *K* must be large enough so that each cluster is representative of each relevant object, yet small enough so that each relevant object is still identified. Thus *k* determines the size of relevant objects although we base the size of individual objects on the distance between SIFT feature descriptors [8]. We treat the problem of choosing *k*, as a function of *n*, as a subtask of clustering. Through several trial and error tests we verify that the following function for k,

Equation 1

$$k = \sqrt{n/2}$$

serves as a tolerable value for *k* when we adjust the initial settings of the SIFT algorithms. Note that determining optimal settings for SIFT requires calibration of filters which is not within the scope of this task but we consider it a future task.

The Distance Function and Randomness:

The original implementation of K-Means assigns *n* k-means points to *k* clusters. Generically a k-means point simply consists of a (x, y) coordinate or in the case of 3-dimensional mapping an (x, y, z) coordinate. The modified k-means object stores a list of k-means points, which contains a SIFT feature. After the initialization of k-means the algorithm randomly assigns a single k-means point to each cluster. Because the k-means algorithm assigns each k-means point to a cluster while updating the entire structure, a distance function is necessary to determine how closely points that do not belong to a cluster relate to the mean of each cluster. The modified k-means algorithm uses a distance function that operates on the SIFT feature of

each k-means point. This value does not identify an exact match, but a probability that there is a match. Thus we also identify a tolerance level that defines how high the probability must be in order for us to consider two SIFT features to match.

Given two k-means points P_1 and P_2 given their feature descriptor vectors P_1d and P_2d, we use the following equation to calculate their distance:

Equation 2

$$\sqrt{\sum_{0 \le i < 128} f(i)}$$

$$f = (P_1d[i] - P_2d[i])^2$$

Clusters as Objects

Because we desire to recognize relevant objects, in modified-k-means we require each cluster to be a representation of that object. Although objects in 2D images suffer from the loss of 3D projection, they still occupy an area of the 2D image. For this reason, we require that a cluster be capable of computing its area. Clusters in the modified-k-means algorithm are able to accomplish this by keeping track of the highest and a lowest value of the Cartesian coordinates of the SIFT features that compose them. More specifically, the cluster uses the maximum and minimum (x, y) values over the composing SIFT features in order to compute a bounding box that contains each point in the cluster. This allows the cluster interface to provide the area that it occupies in the image.

When mapping new clusters to the clusters in the local cache, we can now compare the area of clusters, and match each feature of the new clusters to those of the clusters in the cache. Thus the cluster interface provides a method to obtain all of the features that compose it. We can use the number of matches between clusters SIFT features to identify a relationship between them. It is important to note there is no guarantee that a cluster has more than one point so a cluster containing a single point has a cluster area of zero. This is more likely to occur when

features are sparse or very spread out. A more efficient method of calculating the area that a cluster occupies may enhance the performance of motion tracking using SIFT feature clusters. If we determine that this is valid, then we consider it in chapter six when we discuss future work.

Cluster Validation

Visual image comparison is used for images. A test suite has been implemented that when given a number of files n and n file names, generates a set of images. This set of images includes an image displaying the original set of SIFT features as circles with orientation, the same image showing only the cluster means, and a similar image showing the cluster means when outlier detection is applied. From this set of images we can observe how accurately the cluster means map to the centroid of the object we are identifying. In all of our test cases the output images showed that we were able to find a cluster for each object to be recognized. We can use this to verify that our *k* value is sufficient as the number of objects recognized changes with the equation for *k*. The clusters that we plot do not always align directly with the objects centroid; however with the use of outlier detection they are considerably closer.

Irrelevant Objects and Outlier Detection:

Outlier detection is essential when working with SIFT features and k-means points because we may not know the initial settings of SIFT. Thus it is possible that many of the SIFT features are simply light reflections or some other irrelevant object. Removing irrelevant objects allows us to generate a cluster that maps more closely to the original object. Because the number of outliers is also a function of the number of features to be clustered, outlier detection must be done during the k-means initialization. We implement two types of outlier detection, nearest neighbor detection (NND) and top 50% detection, T50D [9]. Note that for test purposes, we use the algorithm without outlier detection (WOD). The algorithm implemented for outlier detection in this modified k-means maps the distance (see Figure A) of each SIFT feature to its nearest neighbor. The distance and neighbor are stored in a table that is later used to remove a percentage *p* of the number of features. Features are removed in order of the furthest distance from their nearest neighbor. The modified feature set is used when k-means is executed. Alternatively a second outlier detection method is provided which calculates the average nearest

neighbor distance of each SIFT feature. The algorithm then removes all features from the set where the nearest neighbor distance is greater than the average nearest neighbor distance.

Caching and Obstructed Objects

When identifying objects in a sequence of images, moving objects may obstruct stationary or other moving objects between images. Even with outlier detection lighting reflections and refractions can give a misrepresentation of the object. In these situations we compare the area and centroid of existing clusters to determine if one object obstructs another in an image. We use the relationship between the number of points in a cluster, the relative location, centroid, and the overall area size to identify obstructed objects or the best fit. When the identification of previous object is questionable we cache both objects until we identify another object from a more recent image as a match to either of the objects in the cache. In this case we compare all of the objects with matching features and store the object with the largest area and K-Means-Point ratio then remove the previously stored objects.

During the phase in which we compare clusters to identify matches we can also identify the best-fit match. We assume that clusters with more points in a smaller area are more likely to be an accurate representation of an object. We assume the opposite about larger clusters with sparse features. When comparing clusters, if more than one cluster matches the same number of points, then we compare the area of the matching clusters. We identify the cluster with the closest area and the same number of points as the match. We are unable to validate these assumptions without testing.

Motion Tracking

We implement a motion tracking algorithm assuming that because SIFT features are scale invariant, identical features can be identified from a 2D image taken at time t to one taken at time $t + n$ where n is the amount of time necessary to take and process a single image. Therefore, when processing new clusters with matches we can use the Cartesian location of the clusters centroid location in the image to determine a direction of travel in one of eight directions: North, South, East, West, Northeast, Northwest, Southeast, and Southwest. By obtaining the robots current position and orientation we can determine the orientation of the 2D image by rotating the x and y-axis accordingly. We use the mean of the new cluster and the matching cluster to

calculate the slope of the line connecting the two points. The direction is a function of the maximum value of the slope equation. If both the rise and run are equal to zero we identify the object as stationary.

CHAPTER 4 - Experimentation

Robot Image Processor (RIP) is an application that can communicate with mobile robotic device through a socket connection (the connection occurs in the main method of the RIP). The RIP implements both the modified K-Means and SIFT feature identification and extraction algorithm in order to generate clusters of SIFT features. RIP also implements TakePhotoAndProcess, which is a command that sends a message to an agent through the socket connection to take a photo and send it back. When RIP receives an image it generates and processes a set of SIFT cluster features to track the motion of objects.

Now that we are able to clearly identify the task, objective, and necessary algorithms we begin to construct a system to track motion using SIFT feature point clusters. First we identify a test scenario in which the system we develop can be useful. Next we identify and describe any components necessary to execute the experiment in the given scenario. Then we describe the initialization and operation of the system during that scenario. Finally we execute the experiment under the constructs of the scenario and collect any output.

Recall that in the introduction we identify motion tracking as an important task in robot vision. We choose to perform the experiment in a scenario in which a mobile robotic agent must navigate from one end of hallway in the basement of Nichols to the other. At the opposite end of the hallway there is a person traveling toward the agent. The person travels in an indirect path varying its movement in several directions never obstructing the agent.

Figure 5: Images taken by the agent during the experiment

Such a scenario requires several components in order to perform the experiment including a person, mobile robotic agent, robot controller, and a camera. In this scenario we can use person of any size although in future test we may evaluate the performance of the system using people of various sizes. The mobile robotic agent component is an agent that can navigate from one end of the hallway to the other. In this experiment we use a Pioneer 3 or P3-AT mobile robot with *Player* robot server. *Player* robot server is a commonly used robotic controller. A remote machine uses *Player Joy*, a tool for connecting to and manually controlling the movements of a robotic device using *Player*. We use this to navigate the P3-AT down the hallway. The agent also requires a camera in order to generate the sequence of images for the system to process. We mount and connect the camera to a base extension of the unit to increase its range of view. Note that the agent uses additional software *gPhoto* in order to capture and save images such as those in Figure 5.

Figure 6: Images after SIFT feature generation

Initially, we position both the agent and person at opposite ends of the hallway. We power on the agent, and connect to it using *Player Joy* on a remote machine. The remote machine also instantiates an instance of an RIP and specifies the type of outlier detection to use (if any) when generating clusters. The remote machine then establishes a connection to the agent, and begins invoking the RIP's TakePhotoAndProcess method until the user terminates the system. When the agent receives a "Take Photo" message from the RIP it takes a photo, saves it locally, and transmits it back to the RIP.

When the RIP receives an image from the agent it saves it to the host machine and extracts SIFT features from the image. For testing purposes and further analysis, it also stores the grayscale image with red circles indicating the location of SIFT features (Figure 6). After

extracting SIFT features, depending on the type of outlier detection the user specifies (if any) the system removes outliers and then generates a set of SIFT feature clusters. As with SIFT results, the RIP also saves a grayscale image with red circles indicating the location of the centroid of a cluster (Figure 7).

If the image is the first in the sequence, the RIP places all of the clusters in the system cache. The RIP compares all clusters from succeeding images to those in the caches to detect matches. When a new cluster matches one in the cache, the RIP performs further analysis to check for obstructed objects, in order to determine the best-fitting representation of an object in the original image. The process repeats until the user terminates the system.

Figure 7: Mean cluster features

CHAPTER 5 - Results Analysis

As the system executes, it collects and prints statistical information about the clusters it identifies in an image and their direction of travel. We term an entire set of this statistical information collected by the system over a sequence of images "Matched Object and Directional Information" (MODI). In this chapter we first examine the content of MODI and specify its purpose. Then we identify and define a set of corresponding metrics used to evaluate and measure the performance of the system. Using these metrics we generate result tables to evaluate the performance of the system WOD, with NND, and with T50D. Finally, we compare the overall performance of all three tests to conclude which method (if any) performs best.

processing SIFT ...Tracking images with top % outlier detection

The current file is /users/rfoster/RobotPhotos/IMG_1.jpg

Feature at: X: 762.35614 Y: 476.97754, Scale: 9.919196
Feature at: X: 893.14484 Y: 752.52203, Scale: 9.479965
Feature at: X: 861.74023 Y: 722.89526, Scale: 8.935663
Feature at: X: 1079.2844 Y: 609.95746, Scale: 8.90188
Feature at: X: 1038.0438 Y: 657.8557, Scale: 8.65696
Feature at: X: 1154.7716 Y: 846.36304, Scale: 8.02227
Num Features: 6
processing SIFT ...Tracking images with top % outlier detection

The current file is /users/rfoster/RobotPhotos/IMG_2.jpg

Printing match table for new cluster 0:

New Cluster	Old Cluster	New Cluster Points	Old Cluster Points	New Cluster Area	Old Cluster Area	Num Matches
0	0	21	3	1417667.0013625026	1486601.7438147739	0
0	1	21	25	1417667.0013625026	1668036.3999028355	5
0	2	21	2	1417667.0013625026	904334.7828456201	0
0	3	21	39	1417667.0013625026	1832578.578060329	0
0	4	21	2	1417667.0013625026	773997.8589953855	0
0	5	21	1	1417667.0013625026	363626.7556115985	0

Printing match table for new cluster 1:

New Cluster	Old Cluster	New Cluster Points	Old Cluster Points	New Cluster Area	Old Cluster Area	Num Matches
1	0	13	3	1273162.2636902332	1486601.7438147739	0
1	1	13	25	1273162.2636902332	1668036.3999028355	0
1	2	13	2	1273162.2636902332	904334.7828456201	0
1	3	13	39	1273162.2636902332	1832578.578060329	24
1	4	13	2	1273162.2636902332	773997.8589953855	0
1	5	13	1	1273162.2636902332	363626.7556115985	0

Identified old cluster 3 as a match to new cluster 6, with 24 match points.
The old cluster area is 1832578.578060329 the new area is 1273162.2636902332.

The object appears to be moving in a Southeasterly direction
***

8	4	100.0%	1273162.2636902332	1832578.578060329	SE

Figure 8: Sample MODI

MODI & Tracking Tables

MODI consists of three components, Cluster Identification Information (CII), Cluster Match Information (CMI), and Direction of Travel Information (DTI) corresponding to the SIFT, k-means, and tracking components respectively. CII consists of the location, scale, and orientation of the SIFT feature representing the centroid of a cluster. We represent the CMI as a table mapping the values of a newly identified cluster to those of each cached cluster. The values mapped in CMI include an identification number, cluster area, number of points, and number of matched points. The DTI is a text representation of the direction of travel of an object in motion. We generate CII on a per image basis, the remaining components we generate once per cluster, per image (Note that if there is a match, we d not generate DTI).

We use MODI to construct tracking tables similar to those in Table 2: Tracking WOD. A tracking table consists of a set of initial clusters and cluster area, and for each consecutive pair of images in the test sequence: secondary cluster id, previous cluster id, point percentage match, secondary cluster area, previous cluster area, direction. The default value for any field with an integral type is 0 and ST for the directional field to indicate stationary objects. We place purple borders around some cells with values of 100% point percentage match. This indicates that the cluster in the local cache has more features that match than the new cluster has features. Later we describe how to use tracking tables to construct result tables.

Table 2: Tracking WOD

		Tracking Without Outlier Detection						
Images Mapped	Secondary Image Clusters	Previous Images Cluster Match	Point Percentage Match	Secondary Cluster Area	Previous Cluster Area	Direction		
Img_1.JPG - Img_2.JPG	9	0	0.00%	1289148.129	0	ST		
	10	0	0.00%	524721.2529	0	ST		
	11	1	33.30%	822474.6413	848912.8892	N		
	12	3	100%	1211043.882	1743969.795	NE		
	13	7	47.50%	1455674.353	1576513.327	N		
	14	0	0.00%	931725.2415	0	ST		
Img_2.JPG - Img_3.JPG	15	0	0.00%	992240.2105	0	ST		
	16	0	0.00%	1439457.503	0	ST		
	17	0	0.00%	1260488.926	0	ST		
	18	3	61.29%	1742101.103	1743969.795	N		
	19	0	0.00%	1304848.581	0	ST		
	20	0	0.00%	1004934.512	0	ST		
					0			
Img_3.JPG - Img_4.JPG	21	13	26.67%	1512294.166	1455674.353	NW		
	22	8	33.33%	1703123.986	1789060.752	SW		
	23	0	0.00%	1265262.671	0	ST	**Initial Clusters**	
	24	0	0.00%	1708951.813	0	ST	Cluster Number	Cluster Area
	25	0	0.00%	814704.0921	0	ST	1	848912.889
	26	20	33.33%	675950.3761	1004934.512	N	2	1805163.6
							3	1743969.8
Img_4.JPG - Img_5.JPG	27	26	26.67%	1434219.515	675950.3761	NW	4	798774.316
	28	3	55.17%	1643201.686	1743969.795	SW	5	816507.109
	29	16	25.00%	1440033.035	1439457.503	NW	6	1784019.62
	30	13	62.50%	1020143.448	1455674.353	N	7	1576513.33
	31	15	100.00%	640467.8309	992240.2105	NW	8	1789060.75
	32	0	0.00%	1546505.915	0	ST		

Metrics

To identify a set of metrics that can be interpreted from MODI, we recall the objective of the task. In order to track an object we must first be able to identify that object. Similarly, motion tracking implies the identification of changes in location. Thus to properly evaluate the system we must identify at least two metrics. One used to measure the identification of objects and the other to measure the identification of location changes of those objects. To do this, we first identify comparable qualities of clusters that can be extracted from the MODI, the number of feature points, cluster area, and direction of travel. Using these values we identify four metrics to describe the performance of the overall system, including cluster frequency match rate (CFMR), cluster volume match rate (CVMR), cluster area variance (CAV), and direction prediction accuracy (DPA).

CFMR CMVR & CAV

For each metric we define what it measures and explain how to calculate it. CFMR is used as a measure of object identification and indicates how frequently we are able to identify objects from one image to the next. We calculate this value as a percentage by dividing the number of clusters with matches by the total number of clusters the system generates in an image, and multiplying the result by one hundred. CVMR is a measure of object identification and indicates the percentage of SIFT feature points that compose a new cluster relative to the number of SIFT feature points in the matching cluster. Because the number of feature points in the matching cluster may be greater than those in the new cluster, it is possible that this value may be greater than 100%. In this case we evaluate it as a normal 100% match rate and also mark it for further observation. Average cluster area distance indicates how closely the dimensions of a new cluster relate to those of the matching cluster. CAV is also a measure of object identification and indicates how closely the dimensions of newly identified cluster and a matched cluster relate. To calculate CAV we compute the average of the sum of the difference in area for each matching cluster pair.

DPA

DPA is used as a measure of accuracy for the identification of location changes. To calculate DPA we generate a system of measurement. We design a system to evaluate how frequently newly identified objects that match to a cached object are predicted as moving in the correct direction. We evaluate the system on a 5-point scale, and because there are eight possible prediction directions we develop a prediction accuracy scoring system as follows. An incorrect prediction receives a score of 0 while a correct prediction receives a score of 5. Partially correct predictions such as N vs. NE or W vs. SW receives a score of 2.5. We define the predicted direction as the average weighted direction for each cluster over a given image. The weighted direction value is an integer value calculated as the quotient of the number of points in a new cluster divided by the number of points in the matching cluster.

Example:

Cluster	Match Rate	Direction
1	50%	NE
2	25%	NW

3	75%	SE	
4	0%	ST	
5	0%	ST	(Note: ST indicates a Stationary Object)

In the example, 5 clusters are identified and 3 are in motion. Cluster 1 had a 50% match rate and a direction of NE, cluster 2 had a 25% match Rate and a direction of NW, and cluster 3 have a 75% match rate and a direction of SE. We assign directional values as follows:

Direction	Weighted Values	Value Sums
N	50, 25	75
E	50, 75	125
S	75	75
W	25	25

We use the value sums for each direction to calculate the predicted direction of travel by taking the highest combinations of North South and East West directions. When opposite directional values are present we use the absolute value of the difference of their value sums to choose. In this case the values of N and S are equal to 75 so they are negated and do not include North or South in our predicted direction, the overall value of the East West combination is 125 – 25 = 100 for the value sum of E. So east is the predicted direction.

Interpreting Metrics

Now that we have a set of metrics for analysis purposes we consider how to interpret the results relative to performance. Following, we use the metrics to analyze the performance of the system. We begin with CFMR and CVMR, for both metric higher values indicate better performance in object identification. Lower CAV values are indicative of lower variance in the area of matched clusters indicating better performance. Because DPA value is an integral type over a 5-point scale, the average percentage would be computed by taking quotient of the average DPA divided by 5. Higher DPA values indicate better performance in identifying location changes.

Performance Analysis & Result Tables

To analyze the performance of the system WOD, with NND, and with T50D we construct result tables for each test. Result tables are constructed of values collected from the tracking tables we describe in section 5.1. For each pair of consecutive images in the sequence of test

images, the result table contains a column for each of the defined metrics. Additionally, the result table provides a column for the predicted direction and the actual direction so that we may better observe the behavior of the systems motion tracking predictions. The overall values for CFMR, CVMR, DPA, and CAV for a given test are calculated as averages over all of the images in the set. Next we consider the analysis process applied for each table.

For each metric or column in a result table excluding DPA, we observe the minimum and maximum values for correlations to other metrics or image pairs. DPA is excluded from maximum and minimum comparison because it is not a measure of object identification as the others. We have not yet verified that either minimum or maximum DPA values are comparable to those of the remaining metrics. Similarly, for each metric we identify patterns and duplicate values across result tables. Such patters or duplicate entries could be indicative of performance issues, errors in algorithms, system flaws, or learned behavior. Finally, we search for trends in the averages for each metric. The systems performance WOD provides baseline measurements in each category with respect to our performance metrics. Thus the following NND and T50D result table analysis indicates increases or decreases in performance when in comparison to the WOD table.

Performance WOD

Observing the CFMR values of this result table we first note that the minimum value occurs in the second image pair while the maximum value occurs in the sixth. Neither the minimum or maximum values correspond to those remaining metrics, nor do we identify any visible patterns over the image pairs. The maximum CVMR value occurs in the fifth image pair and corresponds to the maximum CAV value. The minimum CVMR value occurs in the third image pair corresponding to the minimum CAV. There appear to be no visible patterns in the change of CVMR over each pair of images. DPA values do not appear to follow any recognizable patterns at this time; however predicted direction follows a counterclockwise pattern from one image pair to the next.

Table 3: WOD result table

		WOD Results				
Image Pair	**CFMR %**	**CVMR %**	**CAV**	**Predicted Direction**	**Actual Direction**	**DPA Value**
Img_1.JPG - Img_2.JPG	50.00%	60.27%	226734.368	N	SE	0
Img_2.JPG - Img_3.JPG	16.67%	61.29%	1868.691739	N	S	0
Img_3.JPG - Img_4.JPG	50.00%	31.11%	157180.238	NW	SW	2.5
Img_4.JPG - Img_5.JPG	83.33%	53.87%	323785.9215	NW	SE	0
Img_5.JPG - Img_6.JPG	33.33%	70.84%	682123.481	SW	SW	5
Img_6.JPG - Img_7.JPG	100.00%	52.11%	462053.621	SE	SW	2.5
Img_7.JPG - Img_8.JPG	83.00%	61.16%	392960.859	SE	SE	5
Average CFMR	**59.48%**					
Average CVMR	**55.81%**					
Average CAV	**320958.1686**					
Average DPA	**2.142857143**	**43%**				

The average CFMR in the WOD result table indicates that 60% of the time we are able to map clusters from proceeding images to clusters identified in succeeding images. Out of the 60% of the matched clusters we are able to identify the 56% of the features composing the cluster as indicated by the average CVMR percentage. Similarly, a DPA average of 2.14 or 43% indicates how often we predict the correct direction of travel among the 60% of clusters with a match. The CAV average indicates nothing in the base performance evaluation WOD because we have nothing to compare it to.

Performance NND

Result tables for system performance with NND show that the minimum CFMR of 40% occurs in the second image pair and does not correspond to other metrics. The maximum CFMR of 100% occurs in the seventh and last image set which corresponds to the maximum CVMR value of 97.14%; this could imply a relationship between the two metrics. There is no correspondence of the minimum CVMR value of 42.38% is located in the fifth image pair. CVMR values do not follow any visible patterns. The maximum CAV value is 618045.09 occurs in the sixth image pair and the minimum CAV is 92590.31 occurs in the third image pair. CAV values appear not to conform to any patterns.

Table 4: NND result table

		NND Results				
Image Pair	**CFMR %**	**CVMR %**	**CAV**	**Predicted Direction**	**Actual Direction**	**DPA Value**
Img_1.JPG - Img_2.JPG	60.00%	73.22%	553507.19	SE	SE	5
Img_2.JPG - Img_3.JPG	40.00%	64.29%	157543.13	S	S	5
Img_3.JPG - Img_4.JPG	60.00%	57.42%	92590.31	SW	SW	5
Img_4.JPG - Img_5.JPG	80.00%	83.33%	387174.04	S	SE	2.5
Img_5.JPG - Img_6.JPG	80.00%	42.38%	453818.91	NW	SW	2.5
Img_6.JPG - Img_7.JPG	83.00%	74.30%	618045.09	E	SW	0
Img_7.JPG - Img_8.JPG	100.00%	97.14%	586198.66	S	SE	2.5
Average CFMR	**71.86%**					
Average CVMR	**70.30%**					
Average CAV	**406982.4757**					
Average DPA	**3.214285714**	**64%**				

Average CFMR indicates that 72% of the time we were able to map clusters from proceeding images to clusters identified in succeeding images. Out of the 72% of the matched clusters we were able to identify the 70% of the features composing the cluster as indicated by the average CVMR or overall match percentage. Similarly, A DPA average of 3.21 or 64% indicates how often we predict the correct direction of travel among the 72% of matched clusters. We measure average CAV values measured in units of distance measurement and evaluate it in section 5.4 when we evaluate system performance over all metrics.

Performance T50D

System performance with T50D table shows that the minimum CFMR of 25.0% occurs in the third image pair and corresponds to the max CVMR value of 100% (Note that CVMR is 100% at the first and third image pairs) and minimum CAV of 363381.79 units. The maximum CFMR of 100% occurs in the fourth image and does not correspond to other metrics. There is no correspondence of the minimum CVMR value of 36.01% located in the fifth image pair. CVMR values do not follow any visible patterns. The maximum CAV value is 788386.28 occurs in the second image pair and does not correspond to any other metrics. From the result table it appears that CAV values do not to conform to any patterns or trends.

Table 5: T50D result table

		T50D Results				
Image Pair	**CFMR %**	**CVMR %**	**CAV**	**Predicted Direction**	**Actual Direction**	**DPA Value**
Img_1.JPG - Img_2.JPG	50.00%	100%	644311.68	SE	SE	5
Img_2.JPG - Img_3.JPG	50.00%	53.13%	788386.28	S	S	5
Img_3.JPG - Img_4.JPG	25.00%	100%	129454.592	S	SW	2.5
Img_4.JPG - Img_5.JPG	100.00%	56.24%	418469.95	S	SE	2.5
Img_5.JPG - Img_6.JPG	75.00%	36.01%	363381.79	N	SW	0
Img_6.JPG - Img_7.JPG	40.00%	66.67%	400789.61	SE	SW	2.5
Img_7.JPG - Img_8.JPG	75.00%	64.81%	450079.32	NE	SE	2.5
Average CFMR	**59.29%**					
Average CVMR	**68%**					
Average CAV	**456410.4603**					
Average DPA	**2.857142857**	**57%**				

Average CFMR indicates that 59% of the time we re able to map clusters from proceeding images to clusters identified in succeeding images. Out of the 59% of the matched clusters we are able to identify the 68% of the features composing the cluster as indicated by the average CVMR. Similarly, A DPA average of 2.85 or 68% indicates how often we predict the correct direction of travel among the 72% of matched clusters.

Performance Averages & Observations

In this section we analyze the results of each metrics average value over all three tests. Next we compare the findings of test with NND and T50D to test WOD. To visually analyze the average performance of each metric over all 3 tests we have constructed an averages table.

Table 6: The averages table

		Result Averages		
Detection Type	**% Clusters Match (Frequency)**	**% Average Match (Volume)**	**Average Cluster Area Difference**	**Weighted Direction Value**
Without Outlier Detection	59.48%	55.81%	320958.1686	43%
Nearest Neighbor Detection	71.86%	70.30%	406982.4757	64%
Top 50 % Detection	59.29%	68%	456410.4603	57%

We construct Table 6: The averages table, by extracting the averages for each metric from the result tables. The table shows the average CFMR, CVMR, and DPA as percentages for each test, and the average CAV in units.

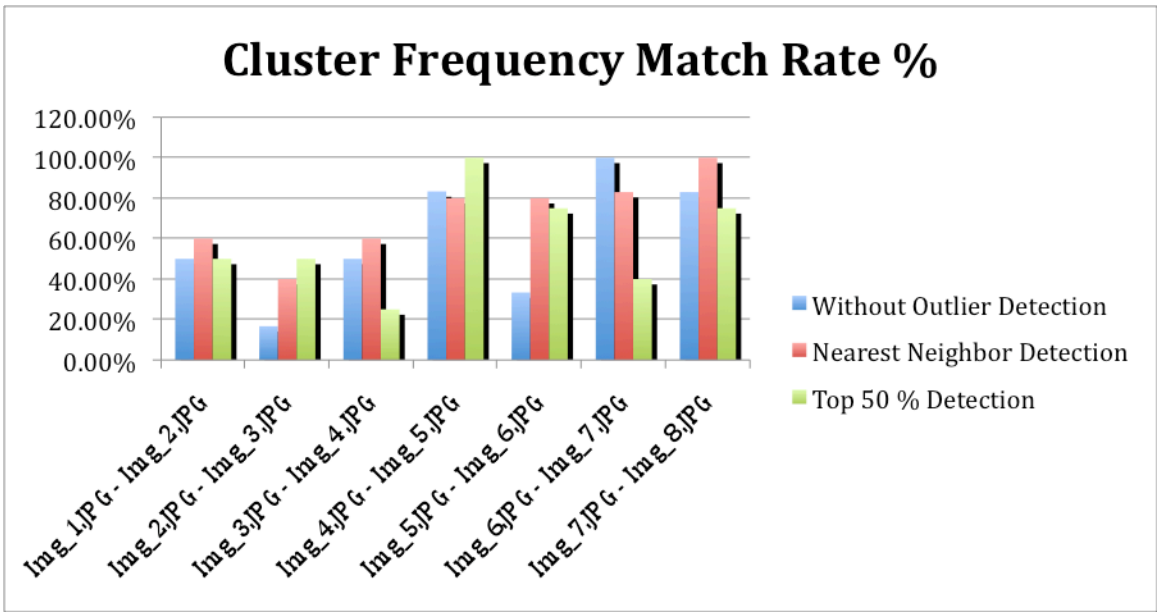

Figure 9: The CFMR graph

From the averages table we see that the lowest CFMR value occurs in the test using T50D, and the highest occurs testing with NND. The minimum value 59% matches the median value, which is also the baseline. Test with NND show maximum value of 72%, a 13% performance improvement in CFMR when compared to test WOD and with T50D. We observe that over all three tests, the maximum CFMR reaches 100%. This indicates that when the same objects appear over a sequence of images the system is capable of generating a set of clusters that map to an existing set. The minimum CFRM values for the test WOD and with NND both occur in the second image pair. The improved performance with NND shows an increasing CFMR over time with the exception of image pair 2. This could indicate that as more images are processed the system is able to recognize objects more frequently. This is the expected behavior for a well performing system, but we cannot verify this without further testing.

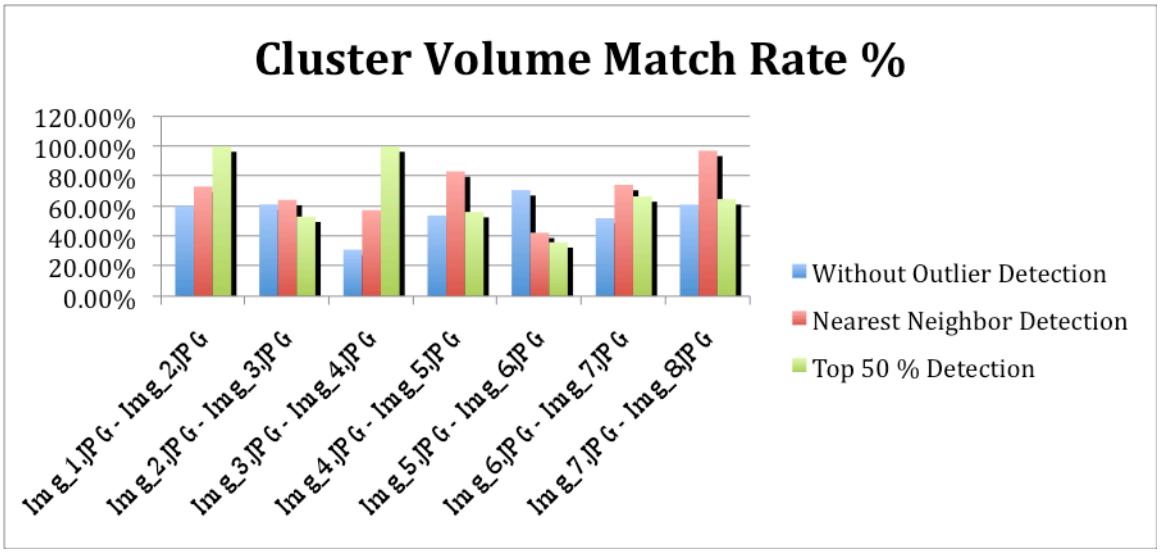

Figure 10: The CVMR graph

The averages table also indicates that using either form of outlier detection improves the system performance measured by CVMR 13% on average, and up to 14% in test using NND. The maximum CVMR indicates that test with NND perform best. In Figure 10: The CVMR graph we see that the minimum CVMR occurs in the fifth image pair for both test with NND and T50D, while test WOD shows its highest CVMR at the same location. This is indicative that between the set of images both outlier detection algorithms treated features of an obstructed object as outliers causing a less accurate match. Further analysis of Img_5.JPG and Img_6.JPG of our test data confirms that the person in Img_5.JPG is obstructing the bulletin board that is later completely visible in Img_6.JPG.

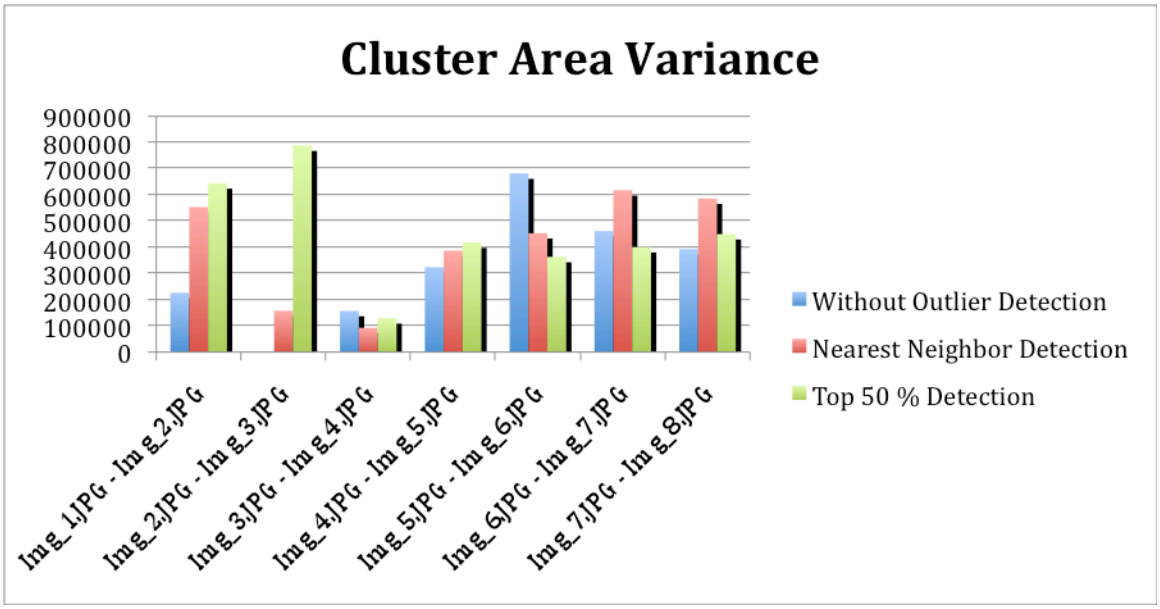

Figure 11: The CAV graph

In the averages table in Figure 5 shows the min CAV value occurs in test WOD and the max in test with T50D. Recall that lower CAV values are indicative of better performance because we consider smaller clusters with more features more accurate. In previous analysis we have seen that test with NND generate the best performance results yet fall in the middle for this metric. In Figure 8 we see that the lowest CAV values for test with either form of outlier detection occur in the third image pair. Further analysis of the IMG_3.JPG - IMG_4.JPG, tracking tables, and result tables indicate that the motion of objects between the set caused few obstructions so that new clusters have similar dimensions.

Test WOD, generates the baseline DPA value of 43%, followed by T50D with 57% and test with NND at 64%. This increase indicates that outlier detection improves system performance measured by DPA by 18% on average and up to 21%. Further analysis of the predicted directions for test with DPA and T50D shows that NND predicts fully accurate direction more frequently than T50D, which predicts partial directions more frequently. Both algorithms fail to even partially identify the direction of travel correctly over one image pair in the set. In test with NND the worst DPA value corresponds to the largest CAV in image pair 6, and in test with T50D the worst DPA value corresponds to the lowest CVMR in image pair 5. In both cases the actual direction of travel was SW, a combined direction and the predicted values

were single directional values. This is indicative that poor performance in CAV or CVMR can affect DPA.

After analyzing each metric individually we can see that increases in object identification performance indicated by CFMR and CVMR lead to increases in location change identification performance measured by DPA. Furthermore, tests using outlier detection perform better over all four metrics than the system WOD. Although there does not appear to be a direct correlation between CFMR and CVMR, it is clear that outlier detection increases the performance of the later. Testing with NND performs the best over all 4 metrics, and in the category of CFMR improves as more image pairs are evaluated.

CHAPTER 6 - Summary & Conclusions, Applications, and Future Work

In summary of our study, we first specify the type of features that we want to cluster. We choose SIFT features because they are scale invariant, and somewhat immune to the problems relating to object identification. Then, we identify a method to cluster the SIFT features using K-Means, and define the modifications that we must make in order for it work. We also design and implement an application to demonstrate the system that allows us to track motion using SIFT feature point clusters over a sequence of images. Next, we devise a real world scenario and experiment in which we test the system, and finally identify a set of metrics to evaluate the performance of the system.

We conclude that it is possible to identify 3D objects from 2D images using clusters of SIFT features. Also, we believe that it is possible to track the motion of objects using clusters of SIFT features as objects. Our best-case results indicate that we are able to identify a 3D object from 2D images 71.86% of the time, and of the objects we identify we are able to track their direction 64% of the time. With some improvements (to be discussed in future work) and further analysis, it may be possible to identify objects more accurately and improve motion tracking and direction prediction accuracy rates.

Applications

Motion tracking using SIFT features can be useful in many applications of computer vision. In the introduction we state that in Robot vision, path planning, obstacle avoidance, and traffic pattern identification relate to the task of motion tracking using SIFT feature clusters. In this section we consider how to apply this design to each of those tasks. As a result of identifying and tracking the motion of objects, we obtain information that proves valuable for path planning. When a mobile robotic agent navigates through a location it usually plans a path in advance. Some path planners such as Wavefront depend on sonar sensing or laser detection to find a route to the destination avoiding obstacles by generating a map and dividing it into a grid. This method assigns probabilistic values to each cell in order to plan. Using a system such as

Wavefront in conjunction with a predictive motion-tracking algorithm such as the one we define in this thesis, it may be possible to improve path-planning performance.

Obstacle avoidance is a task in which a mobile robotic agent avoids colliding with obstacles. A key task in being able to avoid obstacles is to first identify them which is what we do by generating SIFT cluster features. When the agent identifies an obstacle it must decide what actions to take in order to avoid colliding with it. Often that task is as simple as speeding up, slowing down, or stopping. By identifying the obstacle using SIFT feature clusters and tracking its motion (if any), it is possible that we can improve on actions taken in order to avoid obstacles.

We can identify traffic patterns by tracking the motion of multiple moving objects, which is useful because we can then determine the direction of travel for multiple obstacles at once. If we are able to determine a similar path of travel for multiple obstacles, then we can identify traffic patterns. Identifying traffic patterns allows us to make additional decisions when implementing path-planning algorithms; more specifically we can plan our path based on common traffic flows. This has potential to reduce the amount of resources used to re-plan each time we identify an obstacle. It is also possible that using a system similar to ours with some modification, to identify traffic patterns from 2D images only

Future Work

In future work we consider improvements in the following areas in order to achieve better system performance: outlier detection, SIFT, and testing. We first consider an improved outlier detection algorithm. Calculating the average nearest neighbor distance and removing percentage p from our initial SIFT feature set may lead to performance gains. Thus type of outlier detection is a hybrid combination of the two types we implement in the current system. T50D removes half of the features in the set that may be relatively close in proximity to the other features but be very far away from the others. This indicates that the algorithm is likely to remove corresponding features as well. Alternatively some points may fall within the average because the extreme distances of outliers bring it down so that we may not remove enough features.

A preliminary task in robot navigation and obstacle avoidance applications that use SIFT features is the calibration of filters that find relevant objects from which to calculate those features. Calibrating the filters involves finding acceptance levels for step size, initial sigma,

descriptor size, feature descriptor orientation, and the minimum and maximum scale octaves. An implementation of SIFT that performs on color images may allow us to identify features faster and more accurately using color information in order to identify obstructions. Before deciding to use such an implementation of SIFT we must identify whether or not there are tradeoffs in performance by processing color images. In general a more in depth analysis of the minimal image quality and size necessary to obtain accurate results from any SIFT implementation may be help achieves better performances.

Testing over larger image sets and in different environments may allow us to better evaluate the performance of the system and perhaps identify more metrics. It is possible that system may perform better or worse under some conditions as we conduct the proceeding experiment using a high level of control to minimize invariance. Also, our experiment tests the systems behavior when there is only one moving object in the images, although the agent taking the images is also in motion. We suggest testing the system using objects of various sizes and shapes and in varying quantities in order to obtain a better understanding of how to improve the system's performance.

CHAPTER 7 - References

[1] Data Clustering. (n.d.). Retrieved November 15, 2008, from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/wiki/Data_clustering

[2] K-Means Algorithm. (n.d). Retrieved November 15, 2008 from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/wiki/K-means_algorithm

[3] Scale-invariant feature transform (n.d). Retrieved November 15, 2008, from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

[4] Black, M. *Inferring 3D people from 2D Images* [PDF document]. Retrieved from Lecture Notes Online Web site: http://www.cs.brown.edu/~black/Talks/UAI03mjb.pdf

[5] Barni, H., Garzelli, A., Mecocci, A., Sabatini, L. (2000). A Robust Fuzzy Clustering Algorithm for the Classification of Remote Sensing Images. *Geoscience and Remote Sensing Symposium*, *5.* doi:10.1109/IGARSS.2000.858335

[6] Rui, Xu., Wunsch, D., II. (2005). Survey of clustering algorithms. *IEEE Trans. on Neural Networks, 16.* doi: 10.1109/TNN.2005.845141

[7] Lowe, D. (1999). Object recognition from local scale-invariant features. *Proc. 17th IEEE Intl. Conf. on Computer Vision.* Doi:10.1109/ICCV.1999.790410

[8] Salvador, S., Chan, P. (2004). Determining the number of clusters/segments in hierarchic al clustering/segmentation algorithms. *Proc. 16th IEEE Intl. Conf. on Tools with AI.* Retrieved December 18, 2008 from http://cs.fit.edu/~pkc/papers/ictai04salvador.pdf

[9] Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern Classification second edition.* New York, USA: Wiley.