A Person Following Algorithm for Use with a Single Forward Facing RGB-D Camera on a Mobile Robot

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

---

by
Sean Michael Ficht
May 2012

Accepted by:
Dr. Stanley Birchfield, Committee Chair
Dr. Robert Schalkoff
Dr. Adam Hoover

Abstract

This thesis examines the problem of person following. A person following algorithm can be separated into two distinct parts: the detection and tracking of a target and the actual following of a target. This thesis focuses mainly on the detection and tracking of a target person. For the purposes of this thesis a simple robot control architecture is used. The robot moves to follow the target in a straight line. No path planning is considered when executing robot movement.

This thesis aims to accomplish three tasks. First, the system should be able to track and follow a target when no occlusions occur. The non-occlusion scenarios should consider the target in environments with no other people, environments with other people present at different distances, and environments with other people present at similar distances. The second goal will be to track the target person through brief occlusions. The system should be able to detect when the target has been occluded, register the occlusion, and reacquire the target upon completion of the occlusion. The third and final goal of this thesis is to reacquire the target after a long term occlusion. The system must recognize that the target person has disappeared from the scene, wait for the target to reappear, and reacquire the target upon reappearance.

These goals will be accomplished using a generic person detector realized by a HOG person detector, a specific appearance model based on color histograms, a

particle filter that will serve as an integrating structure for the tracker, and a simplistic robot control architecture.

In the following chapters I will discuss the motivation behind this work, previous research done in this area, the methods used in this thesis and the theory behind them. Experimental results will then be analyzed and discussion concerning the results and possible improvements to the system will be presented.

Acknowledgements

I would like to thank Ninad Pradhan for his contributions to this work. We worked alongside one another on this project and his help is greatly appreciated. I would also like to thank Dr. Stan Birchfield for all his input on this research work.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this section the motivation behind designing a robotic person follower will be discussed as well as a general thesis outline given to guide the reader through this thesis.

1.1 Motivation

Every day we see more and more robot/human interaction in society. Robotic intelligence can be seen in many facets of life from car sensors that tell a person when they are too close to an object to robotic vacuums. The intention behind the study is to design a system that will enable a robot to reliably follow a person in an indoor environment. The motivation behind designing this kind of system comes from the desire to have a robot that will carry items for you. For example, in a hospital setting, nurses and doctors must visit countless patients every day. To tend to a patient a doctor need multiple items. Some of these items include medication, charts, a computer to update patient history, and various other medical supplies. The problems are every patient has a different medication and a different chart, there is

not an accessible computer in every patient's room, and the various types of medical supplies such as bandages and shots can add up to be a heavy load. This means that while making rounds a doctor will have to take multiple trips to a supply closets, a computer, and file cabinets. If a system existed that was able to carry all of these items and follow the doctor around, the doctor would be able to visit more patients in a shorter amount of time. This is just one situation in which the proposed person following system could be of use. One can imagine other situations in which it would be beneficial to not have to make multiple trips from one destination to another simply because a person can only carry so much. This is the motivation behind designing a person following robot.

1.2 Thesis Outline

This thesis outlines the design of a system with the ability to detect, track, and follow a specific human target through varied environments in order to serve as a solution to the motivating example. The thesis will progress from pertinent previous work through design, testing, and conclusions.

Chapter 2 will outline existing methods for both person following and person detection. Chapter 3 will cover the theory behind the methods taken in the design of this person following system. Chapter 4 will cover an in depth discussion of the methods used in order to design the system. Chapter 5 will present, catalog, and analyze how well the system performed on the different test scenarios that must be taken into account when designing a person following robot. Finally, chapter 6 will present conclusions about the system and postulate future improvements that can be made to the system.

Chapter 2

Related Work

Related work is broken into two subsections. The first subsection covers person following. The person following subsection considers algorithms that include tracking and following with a mobile robot. The second subsection considers person detection and tracking without a mobile robot. A significant amount of research has also been done in the area of crowd navigation and navigation planning [9],[10],[25]. However, since crowd navigation is not in the scope of this thesis these methods will not be discussed in depth.

2.1 Person Following

Person following deals with implementing detection and tracking algorithms for use with a mobile robot. The detection and tracking portion of these algorithms process incoming frames in order to detect and track a target person and then subsequently pass the processed information to a robot control architecture which then makes decisions about how to move the mobile robot in order to properly follow the target person.

Chen and Birchfield [4] devised a method based on matching sparse Lucas-Kanade [20][24] features in a binocular stereo system. The algorithm, called Binocular Sparse Feature Segmentation (BSFS), detects and matches feature points between a pair of stereo images as well as between images in the video sequence. The BSFS algorithm can be thought of as a two part algorithm. There is a detection mode and a tracking mode.

In the detection mode, a pair of images is fetched from the stereo cameras. Using the Lucas-Kanade approach, feature points are matched between the set of images and the disparity between them is computed. A left-right consistency check method is used. After matching features, a foreground segmentation is performed to remove features that do not belong to the person being tracked. This is done using the known disparity of the person in the previous frame, the estimated motion of the background, and the computed motion of the person. The Viola-Jones [27] face detector is also used in the detection step. It is used to initialize the person being tracked as well as to increase robustness in detection mode. The results of the matched and segmented features are combined with the results of the face detector to determine if a person has been found.

Once a person has been found, the algorithm goes into tracking mode. The person is tracked using Lucas-Kanade from frame to frame. As the features are lost overtime, the algorithm determines whether or not the person has been lost and will return to detection mode if the person has been determined to have been lost.

The BSFS algorithm performs well and does not require the person to wear a different color from the background. It will also work in environments with clutter. It can, however, be distracted by other objects or people with similar motion and disparity to the person being tracked. The algorithm is not able to handle complete occlusions nor can the algorithm handle a disappearance of the target person.

As far as this thesis is concerned, a second valid method of person following was postulated by Brookshire [3]. In his paper he proposes using Histogram of Oriented Gradient (HOG) features combined with a particle filter to follow a target person. The algorithm breaks down into two areas, detection and tracking.

The algorithm detection is performed using video from a single camera of a stereo pair of cameras. The algorithm uses HOG features trained using linear Support Vector Machines (SVMs). The linear SVMs are trained off-line on positive and negative training images. To calculate the HOG features in an image a 64x128 pixel detection window is used. Scanning for HOG features is done at multiple scales. However, to speed up the process of scanning at multiple scales three steps are taken. The algorithm first calculates an internal histogram [27], [29], then scales the internal histogram as opposed to the image, and finally calculates the HOG features. This speeds up the calculation because the internal histogram is only calculated once and then simply indexed to find values at multiple scales.

Once detection has been done, tracking is performed using a particle filter. The particle filter is implemented to curb the effect of missed detections and false positives. The filtering is performed using a particle filter where each particle is processed using a simple Kalman filter. The state of each particle is $\begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$ where z is determined from the stereo pair of images. A constant velocity model is assumed. To simplify the platform motion the system uses readings from an IMU to allow for updating of the pedestrians state relative to the robot.

Results showed the algorithm to be robust to changes in pose, but no results were shown for full or partial occlusions. It should be noted that since a pair of stereo images were used to calculate depth, as opposed to IR sensors, the robot is able to function in an outdoor environment.

2.1.1 Appearance Based Person Following

A widely used method for person following is based on using appearances to segment the target from the surrounding.

Kwon et al. [14] employ a three part following algorithm. The system first establishes correspondences between pixel displacements and camera pan/tilt angles by constructing a two dimensional lookup table that shows the degree of pan/tilt angle needed to move an image pixel to the center of the camera. The system then uses this look up table to estimate the distance from a single target person to the bisecting point of the line connecting the rotating axes of two cameras while at the same time determining the next pan/tilt angles for the cameras to track a target person in the center of the image. The target tracking is done by making a color histogram of the torso of the person. The system segments the blob of the torso in an image using learned color distribution and calculates the center of mass of the blob. This algorithm is subject to illumination changes that cause a shift in the center of mass of the target. It also assumes that the color of the target torso is different from the background and has no method in place for occlusion handling.

Tarokh and Ferrari [23] use color and shape of the persons clothes for target identification. The system applies an iterative threshold method to automatically select threshold values and remove all objects in the scene that have colors different from the target shirt. The remaining pixels in the thresholded image are then segmented using region growing. Then, shape measures are used to select the region that has a shape closest to the shape of the target. Finally, the system uses the image mass, its center, and their derivatives to control the mobile robot motion. This algorithm runs into issues when people have similar colored clothing and there is no method in place to handle occlusion.

Schlegel et al. [18] propose a combination of a fast color-based with a robust contour-based approach. The color-based portion of the algorithm uses a color histogram to provide regions of interest to the contour-based portion of the algorithm. The contour-based portion then searches for the target by matching extracted contours against an adaptive contour template. This algorithm is more robust to situations in which similarly clothed people are present, but does not discuss occlusion handling.

Sidenbladh et al. [21] present an algorithm that looks for a persons head. The system locates the head of a person by using skin color detection. Then a control loop is used to keep the person centrally located. This is done by adjusting the pan and tilt of the camera as well as controlling the wheels of the mobile robot that the camera is mounted on. The algorithm has trouble when objects or background is present that is similar to the target skin tone, neither does it present a method for occlusion handling. The target must also be facing the camera.

2.1.2 Optical Flow Based Person Following

Another method that has been used to implement a person following algorithm deals with using optical flow information. These algorithms are more robust to similar colors between the target and other objects or backgrounds, but are sensitive to camera vibration and subject to drift.

Piaggio et al. [16] propose one such optical flow based following method. The algorithm first calculates the optical flow. The the optical flow image is thresholded in order to segment the person from the background. The system applies a low-pass filter to discard image pixels that don't belong to a person. The width of the target person is then extracted in order to calculate the persons distance from the robot and

the mobile robot is moved accordingly.

Chivilo et al. [5] use the optical flow calculated in the center area of the image. The optical flow in this algorithm is calculated using the Horn and Schunk algorithm [11]. Then, the basic idea of the system is to measure the relative velocity of the target person with respect to the mobile robot and keeping this relative velocity as small as possible.

2.1.3 Stereo Based Person Following

As seen earlier in work by Chen and Birchfield stereo vision is also used in person following algorithms.

Another example of a stereo based person follower comes from work done by Beymer and Konolige [2]. The algorithm first creates an orthographic floor-plane representation of the three dimensional stereo camera information. The system then subtracts the background in one of two ways. (1) If the robot is not moving, average background subtraction is used. (2) If the robot is moving, the system uses odometry data to estimate background motion which is then subtracted. Once the background has been subtracted, the system finds the target by modeling people as Gaussian blobs. The target is then assigned a state vector and a Kalman filter is applied to maintain the location of the person. No method of full occlusion handling is discussed in this paper.

2.1.4 Other Person Following Methods

Other methods of person following that have been attempted include laser based person following [13], video and RFID based person tracking [7], and tracking based on active contours [26].

2.2 Person Detection and Tracking

Person detection and tracking differs from person following in that the algorithms are not designed for use with a mobile robot. Cameras used are typically stationary which means the background scene is constant.

One successful method for person tracking in RGB-D data was proposed by Luber et al. [15]. The algorithm combines a multi-cue person detector for RGB-D data with an online detector that learns individual target models.

The person detector for RGB-D data is made up of a HOG detector [6] performed on the color image and a HOD (Histogram of Oriented Depths) detector [22] performed on the depth image. This approach is called Combined Histogram of Oriented Depths and Gradients (Combo-HOD). The HOG and HOD descriptors are computed at multiple scales. This calculation is sped up by using integral tensors which is an extension of an integral image. The descriptors are combined using a weighted mean of the probabilities obtained by a sigmoid that is fit to the SVM outputs. The output of the detector are the positions and size of all the targets in 3D space.

The online detector builds upon the online-boosting method for object detection proposed by Grabner et al. [8] who propose applying on-line boosting to 'selectors' rather than directly to weak classifiers, where the 'selector' selects the weak classifier with the lowest error. Luber et al. [15] compute 3 types of features that correspond to the weak classifiers. The three include Haar-like features in the intensity and depth images and illumination agnostic features in the color image. The features are computed in rectangular regions with random positions and sizes within a targets bounding box. This is done once when the target is initially found and then fixed for the duration of the target's presence. On-line boosting is then used

to continuously update the target model. To keep constant, the region for feature detection is the same size as the bounding box of the previous detection. By sweeping the bounding box around a local neighborhood, a confidence map is created yielding the new tracking position at its maximum.

The on-line detector is then brought into a Kalman filter based on Multi-hypothesis tracking framework [17] differing in that their on-line classifier adds an appearance likelihood that calculates how much the observed target appearance matches the learned model.

Finally, at each iteration the tracker produces assignments of measurements to tracks as well as interprets the measurements as being new tracks or false alarms. It also decides if a track has been occluded or deleted.

The algorithm seems to perform quite well. The algorithm is designed for detection and tracking of multiple targets as opposed to tracking one specific target and occlusion is not handled. Once a target has been determined to be occluded the tracking history of that target stops and if that target reappears it is then determined to be a new target as opposed to a target reacquisition.

In another approach proposed by Bansal et al. [1], a method of pedestrian detection based on structure and appearance classification is implemented.

First, a depth map of a given scene is produced by a set of stereo cameras. Pre-computed templates at three separate depth intervals are used at runtime to search for prospective candidates. Using template matching based on the depth interval that is being searched, a correlation score map is produced. Using non-maximal suppression, the peaks or the correlation score map are selected and initial regions of interest for prospective candidates are the result. This initial set is reduced by first considering overlapping regions of interest. If a region of interest overlaps an existing detection by more than 70 percent it is discarded. Next, Canny edges are found for each region

of interest and a vertical projection of the binary mask of the edges gives them a one dimensional histogram. Peaks of this histogram are then detected using mean-shift with each peak corresponding to a possible pedestrian. A new region of interest is then set at each peak with overlapping regions once again being tossed out. At the same time, the algorithm is labeling pixels, based on height, as belonging to either the ground plane, tall vertical structures, overhangs, or pedestrian candidates.

An image based classifier further evaluates the candidates. A HOG feature detector is combined with contour segments of body parts in this classifier. To combine the two methods, the algorithm uses chamfer matched templates on the regions of interest to create a foreground mask. The foreground mask is then used globally on the image to suppress the background giving enhanced gradient values on pedestrian contours during the HOG calculation while suppressing others that are potentially from the background.

This algorithm also performs quite well but once again is most well suited for pure pedestrian detection in a moving car and does not employ any long time person specific descriptors that would be needed to track the same target for long periods of time.

A widely used method in person detection algorithms is Dalal and Triggs' Histogram of Oriented Gradients for Human Detection method [6] which will be discussed in detail later in this thesis. A multitude of algorithms exist that have implemented some variation of HOG or expounded upon it [28], [19], [22] are just a few examples of such algorithms.

Chapter 3

Background Theory

3.1 RGB-D Sensor

First, we need to look at how the RGB-D sensor works in order to validate it as a useful tool for the purposes of tracking. The sensor used is the Xbox Kinect camera. It is made up of two main parts. It has a projector and an IR VGA camera. The projector bounces a laser across the entire field of view of the camera. The camera then picks up the projected laser in order to segment objects into a depth field. The reason this works is because the sensor gets all the pixels back as IR noise measurements that vary in color depending on how close that pixel position is to the camera. This is how the depth image is created and obtained. The RGB image is obtained as it would be in any other camera. So essentially the RGB-D sensor is providing us with a color image and a depth image that can be accessed as often as they need to be. It works at 30 fps.

3.2 Homographies

In order to determine the speed at which our person of interest is walking toward or away from the mobile robot, we must first be able to relate the distance between pixels in the image to that same distance in the real world. In order to do this image coordinates will have to be transformed into real world coordinates. To accomplish this task a homography will be used.

To understand the homography transform, we must first discuss pin-hole camera geometry. The pin-hole camera, Figure 3.1, is defined by its optical center C and the image plane I. The distance from C to the image plane is the focal length and the line from the camera center perpendicular to the image plane is called the optical axis of the camera. The relationship between the 3D world coordinates of a scene point and the coordinates of its projection onto the image plane is described by the perspective projection. A 3D point is projected onto the image plane with the line containing the point and the optical center.

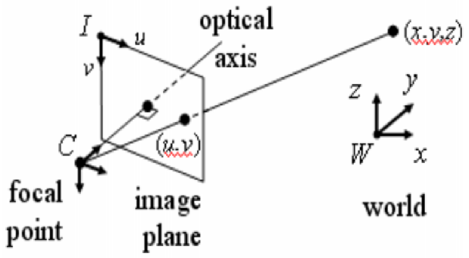

Figure 3.1: Pin-hole camera geometry

Letting the center of projection be the origin of a Euclidean coordinate system, where the z-axis is the principal axis; by similar triangles, Figure 3.2, it is easily seen that a 3D point (x, y, z) is mapped to the point $(\frac{f_x}{z}, \frac{f_y}{z})$ on the image plane.

If the world and image points are represented by homogeneous vectors, then

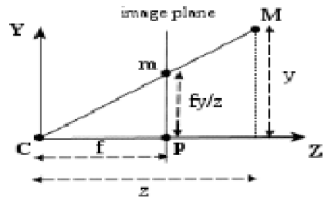

Figure 3.2: 3D mapping to image plane

perspective projection can be expressed in terms of matrix multiplication as seen in the equation below.

$$\begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

The matrix describing the mapping is called the camera projection matrix P. This equation can be written more simply as:

$$zm = PM$$

where $M = (x, y, z, 1)^T$ are the homogeneous coordinates of the 3D point and $m = (\frac{fx}{z}, \frac{fy}{z}, 1)^T$ are the homogeneous coordinates of the image point. Since it only contains information about the focal distance f, the projection matrix P represents the simplest possible case.

This formulation assumes a special choice of the world coordinate system and the image coordinate system. This can, however, be generalized by introducing changes of the coordinate systems. Changing coordinates in space can be done by multiplying the projection matrix P by a 4x4 matrix composed of a rotation matrix

R and a translation vector t. It describes the position and orientation of the camera with respect to the world coordinate system.

$$G = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

The rows of the rotation matrix R are unit vectors that, together with the optical center, define the camera reference frame, expressed in world coordinates. Changing the coordinates in the image plane is equivalent to multiplying the matrix P on the left by a 3x3 camera calibration matrix K.

$$K = \begin{bmatrix} f_x & -f_x cot\theta & u_0 \\ 0 & \frac{f_y}{sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

In this calibration matrix, f_x is the horizontal focal length, f_y is the vertical focal length, θ is the angle between the horizontal and vertical axes, and (u_0, v_0) represents the intersection of the optical axis with the image plane. Thus, the camera matrix in general is the product of three matrices:

$$P = K[I|0]G = K[R|t]$$

Given a world plane and the camera matrix we are now able to formulate our homography matrix. Referring to Figure 3.3, the formulation of the homography matrix is as follows:

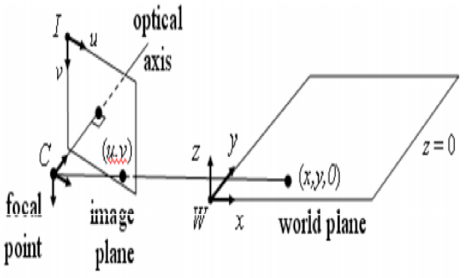

Figure 3.3: Mapping world coordinates to image coordinates

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \begin{bmatrix} f_x & -f_x cot\theta & u_0 \\ 0 & \frac{f_y}{sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix} [R_{3x3} t_{3x1}] \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

$$= K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

In this equation, $K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}$ represents the 3x3 homography matrix, H_{3x3}, that will be used in order to map real world coordinates to image coordinates which will be used in our robotic follower to measure the targets movement distance between frames.

3.3 Histogram of Oriented Gradients

Person detection using Histogram of Oriented Gradients (HOG) was first proposed by Dalal and Triggs [6]. It has become the standard for person detection in RGB and grayscale images. The underlying assumption in HOG is that gradient information is discriminative enough to detect people in an image. We will now describe the OpenCV implementation of Dalal and Trigs' HOG person detector.

The first step in computing HOG features involves computing the gradient of the image. The gradients used are computed using simple $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$ kernels. For color images, separate gradients are calculated for each color channel and the one with the largest norm as the pixel's gradient vector is the one chosen.

Once the gradient has been calculated, a binning procedure takes place. In this step we are calculating a cell histogram. A cell is comprised of an 8x8 pixel area. Each pixel within a cell contributes a weighted vote towards an edge orientation. Orientations are separated into 9 separate bins between 0 degrees and 360 degrees based on the angle of the gradient at the current pixel as seen below in Table 3.1

degrees	bin
0 - 39	1
40 - 79	2
80 - 119	3
120 - 159	4
160 - 199	5
200 - 239	6
240 - 279	7
280 - 319	8
320 - 359	9

Table 3.1: Cell histogram binning procedure

The weighted vote that goes into the bin is based on the magnitude of the gradient at the current pixel. The votes of all the pixels contained within a cell make up the cell histogram.

Once cell histograms have been computed, the cells are combined to form blocks. A block is a 16x16 pixel space, and the blocks overlap each other by 8 pixels as seen below in Figure 3.4. Hence, most cells will contribute to four blocks.

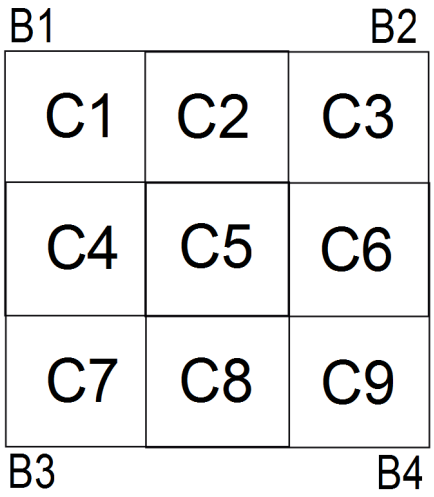

Figure 3.4: Block merging

The reason for merging the cells into blocks is to reduce the effects of illumination variation as well as to reduce the effects of foreground-background contrast. Once the cells are grouped into a block the block is normalized using the L2-Hys norm. The L2-Hys norm is simply the L2 norm followed by clipping, limiting the maximum values of the non-normalized block descriptor vector, $\mathbf{v}$, to 0.2 and re-normalizing. The equation for the L2 norm is as follows:

$$f = \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_2^2 + e^2}}$$

where $\mathbf{v}$ is the non-normalized descriptor vector of a block.

The overall descriptor from a 64x128 pixel detection window is a concatenated vector of all the normalized block descriptors. It should, however, be noted that a 16 pixel margin is included around the 64x128 detection window.

The last step is to send the HOG descriptors into a Support Vector Machine classifier. This is a binary classifier that looks for an optimal hyperplane as a decision function. The classifier is trained on thousands of test images. It is given both positive (person present) and negative (person not present) images before the classifier is ready to be tested. Dalal and Triggs use a soft linear SVM trained with SFMLight [12].

3.4 Particle Filter

The particle filter is a method of approximating difficult (non-Gaussian) probability distribution functions. The particle filter belongs to the family of sequential Monte Carlo methods. Based on Bayes' rule, tracking involves computing the posterior:

$$p(\mathbf{x}_t|\Phi_{0:t}) \propto p(\Phi_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\Phi_{0:t-1}) d\mathbf{x}_{t-1} \tag{3.1}$$

where $\mathbf{x}_t$ is the state at time t, and Φ_t is the measurement obtained at time t. The motion model is given by $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, and the observation model is given by $p(\Phi_t|\mathbf{x}_t)$. The factor $p(\mathbf{x}_{t-1}|\Phi_{0:t-1})$ is the posterior from the previous time step. The Particle filter assumes that the process is Markov, i.e., $p(\Phi_{0:t}|\mathbf{x}_{0:t}) = \prod_{i=0}^{t} p(\Phi_i|\mathbf{x}_i)$.

Then the maximum a posteriori estimate (MAP) is found:

$$\mathbf{x}_t^* = \arg\max_{\mathbf{x}_t^{(i)}} p(\mathbf{x}_t^{(i)}|\Phi_{0:t-1}),$$

for $i = 1, \ldots, M$, where M is the number of particles.

To implement the particle filter, two initialization steps must first take place.

1. Declare the number of particles to be used, M.

2. Initialize all the particles with equal weights.

$$\chi = \{x^m, w^m\}_{m=1}^{M} = \{0, \frac{1}{M}\}_{m=1}^{M}$$

Once this has been done, the following algorithm is run repeatedly until the entire program has finished.

1. Transition each particle with a different random dynamic noise.

$$\{x_t^m\}_{m=1}^{M} = \{f(x_{t-1}^m, \mu_t^m)\}_{m=1}^{M}$$

2. Using the new observation, update each particle weight.

$$w_t^m = w_{t-1}^m p(\Phi_t | x_t^m)$$

3. Normalize all the new weights so they sum to 1.

$$\left\{ w^m = \frac{w^m}{\sum_{m=1}^{M} w^m} \right\}_{m=1}^{M}$$

4. Compute the expected value.

$$E[x] = \sum_{m=1}^{M} x^m \cdot w^m$$

5. If necessary re-sample.

The need for re-sampling arises due to the fact that particles will wander as they are iteratively propagated by the motion model.. As the algorithm progresses, some particles will drift away from the target observation sending their weights closer and closer to zero. As more particles drift, we are left with only a small amount of particles that carry any weight with them and, hence, are the only particles contributing to the calculation of the expected state. This creates a problem of reliability. Therefore a re-sampling criteria is set such that:

$$CV = \frac{VAR(w^m)}{E^2[w^m]} = \frac{\frac{1}{M}\sum_{m=1}^{M}\left(w^m - \frac{1}{M}\sum_{m=1}^{M} w^m\right)^2}{\left(\frac{1}{M}\sum_{m=1}^{M} w^m\right)^2} = \frac{1}{M}\sum_{m=1}^{M}\left(M \cdot w^m - 1\right)^2$$

where CV is known as the coefficient of variation and is used to calculate the effective sample size:

$$ESS = \frac{M}{1 + CV}$$

which lets us know how many particles still have valid weights. If the ESS is less than some predetermined ratio of M then it is determined that a re-sample is necessary. In this thesis the method of select with replacement is used to re-sample.

Select with replacement works by taking all the values of the normalized weights, which are valued from 0 to 1, and indexing them by particle number. The weight can be thought of as the y-axis of a graph and the indices of the particles from 1 to M can be thought of as the x-axis of a graph. Next, we want to create a cumulative weight graph. This means that particle 2 on the graph is equal to the weight of particle 1 plus particle 2. This is done all the way through particle M and particle M's weight is guaranteed to be a value of 1. Finally, for M number of particles, a value between 0 and 1 is chosen at random and the index of the old particle

that holds that value is the index that is assigned to the new particle, meaning the old particle state now the state of the new particle. This ensures that particles with lower weights are less likely to be copied into the new particle list and particles with higher weights will most likely be copied multiple times into the new particle list. The reason that particles with higher weights will be copied more often stems from the cumulative distribution that was set up. It is more likely that the number chosen at random will fall within the range of a particle with a large weight range than it will a particle with a small weight range. A graphical representation of this process can be seen below in Figure 3.5.

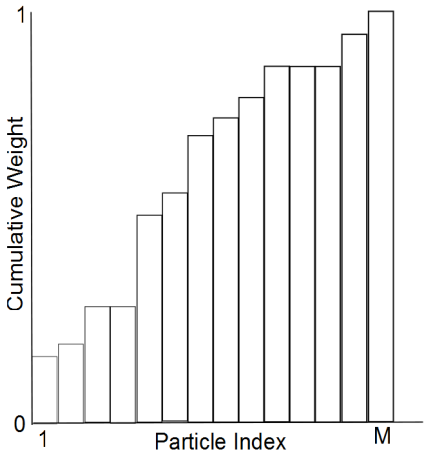

Figure 3.5: Graphical representation of re-sampling

3.5 Color Histogram

A color histogram is a representation of a distribution of colors in an image. It is a way of categorizing the number of pixels in a given image by the RGB characteristics that they display. This is normally done by dividing each color channel into a certain number of ranged bins and then defining the color histogram values to be all the different possible combinations of these ranged bins. For example, dividing each channel into two bins would result in the color histogram seen in Table 3.2.

Red	Green	Blue	Pixel Count
bin 1	bin 1	bin 1	pixels
bin 1	bin 1	bin 2	pixels
bin 1	bin 2	bin 1	pixels
bin 1	bin 2	bin 2	pixels
bin 2	bin 1	bin 1	pixels
bin 2	bin 1	bin 2	pixels
bin 2	bin 2	bin 1	pixels
bin 2	bin 2	bin2	pixels

Table 3.2: Example of a simple color histogram

where bin 1 means that a pixel as a value between 0 and 127 on that channel and bin 2 means it has a value between 128 and 255. It can be seen here that when dealing with a three channel color histogram the number of entries in the color histogram is proportional to the number of bins that are uses to separate the ranges. The number of entry elements in a color histogram can be calculated by:

$$(number\ of\ bins)^3$$

Comparing the color histograms of two separate images provides a way of giving a measure of the appearance likeness between the two images.

Chapter 4

Methods

4.1 Overall Procedure

The algorithm is initialized by the user clicking on the person to be followed during the duration of the experiment. Once this click has been received, the M particle filter particles are initialized and the initial appearance model is constructed. In this implementation 1000 particles were used, $M = 1000$. Then, the algorithm consisting of generic person detections, the specific appearance model, and the integrating particle filter begin to run in order to detect, track and follow the target person through a cluttered environment. The initialization procedure can be seen below in Figure 4.1

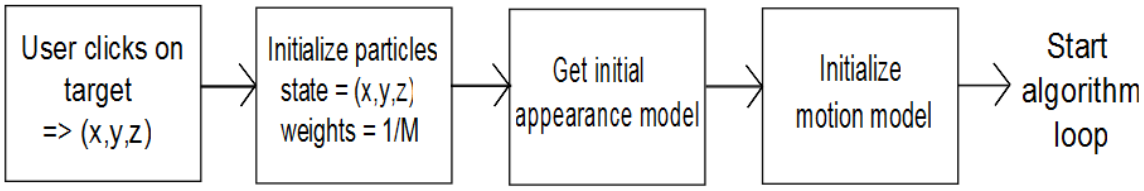

Figure 4.1: Initialization

Color data from the sensor is used to run a generic person detector. The generic person detector used in this thesis is Dalal and Triggs' HOG person detector. It generates gradient images and uses them to define the HOG descriptors which are classified using a linear SVM classifier to provide generic person detections.

The person specific appearance model uses the color data in order to create an appearance model. The purpose of the appearance model is to make it possible to detect occlusions of the target person. If the target person is occluded by another non-target detected person, the generic HOG detector and particle filter alone would not detect the occlusion and would assume that the occluding party is the target person. This more often than not will shift the system into thinking that it is now following the occluding party, thereby deviating from following the actual target. By employing a person specific appearance model, we can avoid this confusion. The person specific appearance model must be robust to minor changes in the appearance of the target due to pose change, but at the same time sensitive to major changes that will set off an occlusion indicator regardless of whether or not a person has been detected in the same area.

Scaled image coordinates and depth information create a hybrid state space for the particle filter which is used for tracking. The person detections generated for the generic HOG detector are compared to the propagated particle locations in the hybrid state space. The weight of the particle is then updated using both the indication of any person in that area and the specific appearance model to determine the likelihood that this is the target person.

The appearance model is then used once again on the tail end of the algorithm to conduct a final check on the target person. If the particle filter expected position matches the specific appearance model within a given threshold then the system logs the expected position as a positive detection. If the particle filter expected position

does not match the specific appearance model within the given threshold then it is determined that an occlusion has occurred and the target position is not updated.

Finally, all the detection and tracking information is sent to the robot control architecture which will determine, based on distance to the target and target position within the frame, how to move in order to follow the target. The overall algorithm procedure can be seen below in Figure 4.2.

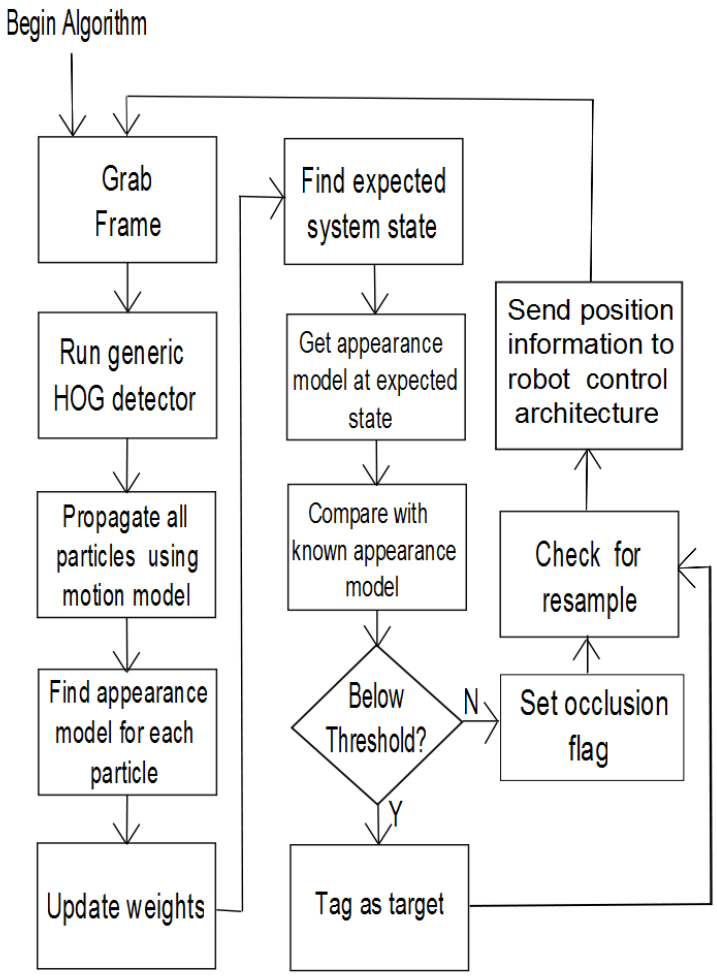

Figure 4.2: Algorithm flow

4.1.1 Overall Methods Pseudocode

1. Initialize
2. Loop:
   (a) Find generic detections (Section 4.3)
   (b) Execute particle filter tracking (Section 4.2)
   (c) Check for occlusion using appearance model (Section 4.4.2)
   (d) Commence robot control (Section 4.5)

4.2 Particle Filter Tracking

4.2.1 System state

The system state consists of three spatial coordinates x, y and z.

$$\begin{bmatrix} x & y & z \end{bmatrix}^T$$

The incoming data gives both an RGB image and a depth image so the coordinates are in a hybrid state space with x and y in image coordinates and z in depth coordinates. These are the state values that will be propagated for each particle and will identify the location of the target person.

The depth coordinate z comes directly from the depth data obtained from the RGB-D sensor. The coordinates x and y are scaled according to the depth coordinate z. The purpose for this is to equalize the weight contribution from each state element.

4.2.2 Motion model

The motion model operates in one of two ways. If the target has been consistently found then the motion model operates in its normal mode. In this mode, the particles are propagated in random directions according to a ray architecture that is set up. At each iteration, 12 separate ray directions are defined in the x and z plane and then using a random number generator, one of the ray directions is chosen for each particle. If you imagine the ray spread in Figure 4.3 as being top down in the world plane then this is how the particle states x and z are propagated.

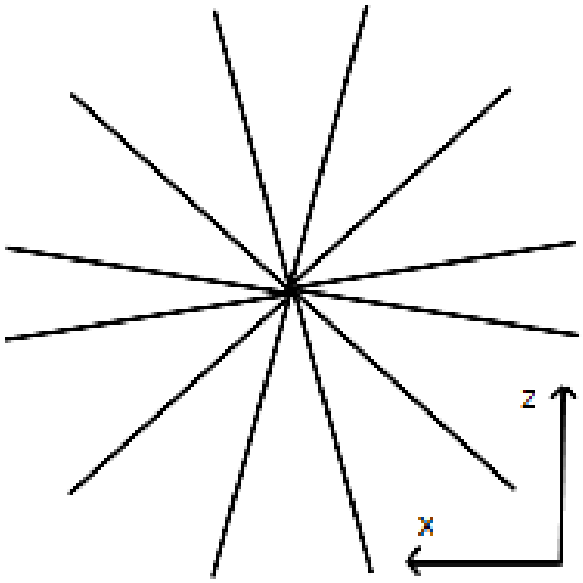

Figure 4.3: Propagation rays

Once a ray direction has been chosen, the propagation of each particle in x and z is then determined by adding the corresponding ray coordinate to the previous x or z state and then adding random Gaussian noise. The y value of each particle is simply propagated by adding a random Gaussian noise to the previous state. The

equations governing this propagation can be seen below in equation 4.1.

$$\begin{aligned} x_t &= x_{t-1} + r_x + N(0, \sigma_x^2) \\ z_t &= z_{t-1} + r_z + N(0, \sigma_z^2) \\ y_t &= y_{t-1} + N(0, \sigma_y) \end{aligned} \tag{4.1}$$

where r_x is the x ray coordinate and r_z is the z ray coordinate. The reason for not including y in this ray scheme is due to the fact that the target will move very little in the y direction throughout a test sequence. The target will mostly be moving in the x, z plane.

One deviation from the simple motion model that has been made comes from evaluating a particles location in x image coordinates. When a particle moves close to image bound, the propagation direction is influenced such that the rays will be directed away from the image boundary and will propagate with a larger magnitude than it generally would if it were located more centrally in the image. This is done for three reasons. First, it is known that the particle must stay within the image bounds and a particle state that lies outside the image bounds does not make sense. Second, the robot control architecture is making sure that, for the most part, the target stays centrally located in the image. Finally, since the tracking is being done indoors in hallways, image bounds are most likely going to belong to walls or the ceiling. Setting these conditions on particle motion simply helps to avoid re-sampling too often.

The second mode of operation for the motion model occurs when the target has not been located for 3 frames. If this is the case, the magnitude of the particle propagation is increased in order to spread the particles across the image more. This was implemented to solve cases in which the particles may drift off the target person and can not reacquire the target because the propagation of each particle is too small to get back to the target location.

4.2.3 Observation model

In order to update the particle weights, each particle state is evaluated against each generic HOG detection in the $\begin{bmatrix} x & y & z \end{bmatrix}^T$ state space. This on its own is not sufficient enough to bias particle weights toward the target person due to the fact that a weight evaluated in state space will be high as long as it is near any person. Thus, the individual weight contributions are scaled by an appearance dependent scale factor. To perform this comparison an appearance model around each particle is constructed. The appearance model of each particle is then compared to the appearance model of the target taken in the initial frame. The comparison is calculated in equation 4.2.

$$\Delta_{d_s} = e^{\frac{-(\|d_s^m - d_s^{orig}\|)}{2\sigma^2}} \tag{4.2}$$

Where d_s^m is the particle appearance model, d_s^{orig} is the initial appearance model and Δ_{d_s} represents their comparison. Therefore, a large difference in the appearance models results in a small scaling factor Δ_{d_s}. The observations for each value in the state space are then compared against all generic HOG detections. The equations for these calculations can be seen below in equation 4.3

$$\begin{aligned} O_{t_x}^m &= \sum_{p=1}^{P} e^{\frac{-(\phi_{t_x}^p - \phi_{t_x}^m)^2}{2\sigma^2}} \\ O_{t_y}^m &= \sum_{p=1}^{P} e^{\frac{-(\phi_{t_y}^p - \phi_{t_y}^m)^2}{2\sigma^2}} \\ O_{t_z}^m &= \sum_{p=1}^{P} e^{\frac{-(\phi_{t_z}^p - \phi_{t_z}^m)^2}{2\sigma^2}} \end{aligned} \tag{4.3}$$

where P is the number of generic detections and $O_{t_x}^m$ is the summed observation resulting from comparing the x state variable $\phi_{t_x}^m$ of the m^{th} particle to each of the x values $\phi t_x{}^p$ of P generic person detections. Here, the generic term O_t is taking the

place of $p(\mathbf{\Phi}_t|\mathbf{x}_t)$ from section 3.4 meaning that $O^m_{t_x}$ would represent $p(\phi^m_{t_x}|x^m_{t_x})$. For each particle these observation equations are a calculated sum that takes all generic detections into consideration. This is done for all M particles.

Once the summed observations for a particle have been calculated, each particle weight is updated by multiplying the previous weight value by the sum of the observations scaled by the appearance difference Δ_{d_s}. Mathematically:

$$w^m_t = w^m_{t-1} \cdot \Delta_{d_s} \cdot (O^m_{t_x} + O^m_{t_y} + O^m_{t_z}) \tag{4.4}$$

From equation 4.4 it can be seen that if a particle state is not close to a generic detection location or the particle appearance model is not similar to the initial appearance model its weight will not be high. After all particle weights have been updated, they are normalized to a value of 1:

$$\begin{aligned} W &= \sum_{m=1}^{M} w^m \\ w^m &= \frac{w^m}{W} \qquad for \quad m = 1 \dots M \end{aligned} \tag{4.5}$$

The normalized weights are then used in order to calculated the expected state which is used for the current target expected position. The expected state is

calculated as in equation 4.6

$$E_x = \sum_{m=1}^{M} x^m \cdot w^m$$

$$E_y = \sum_{m=1}^{M} y^m \cdot w^m \tag{4.6}$$

$$E_z = \sum_{m=1}^{M} z^m \cdot w^m$$

These expected state values are the values returned from the particle filter and indicate the particle filters expected location of the target person. This location will be checked for occlusion by the specific appearance model to determine whether or not an occlusion has occurred. After the occlusion check, regardless of whether or not the target is detected, a check is performed on the particles to determine if a resample is necessary. The equation for determining the necessity of a resample can be seen in section 3.4. If a resample is necessary then it is done using the select with replacement method.

After the resample check, the tracking algorithm is finished and sends its finding to the robot control architecture which responds accordingly.

4.2.4 Particle Filter Tracking Pseudocode

1. Propagate particles
2. For all M
   (a) Find appearance model at particle state (Equation 4.1)
   (b) Compare with initial histogram (Equation 4.2)

(c) Compare particle state to generic detections (Equation 4.3)

(d) Update particle weights (Equation 4.4)

3. Normalize weights (Equation 4.5)

4. Calculate expected state (Equation 4.6)

5. Check for re-sample

4.3 Generic Detector

In this algorithm, the generic detector is used to find all the possible person detections in a given frame. The generic detector used in this algorithm is the HOG person detector. The OpenCV implementation was used. It is a multi-scale detection. In this implementation block sizes are 16x16 pixels, cell sizes are 8x8 pixels and the size of the detection window is 64x128. This means that there are 105 overlapping blocks per detection window. Since the purpose of the generic detector is not to locate one single person, but instead to present multiple detections to the particle filter and specific appearance model, the detection rate for the HOG detector is set high. This means that after the generic person detection portion of the algorithm runs there are many false positive detections. This, however, is not a problem for the purposes of this algorithm, because the false positives will simply be thrown out by the particle filter and/or appearance model. It is not the job of the generic detector to be precise in its classification of people. We simply want to make ensure that there are no false negatives detected. The false positives are dealt with during the consideration as to which target is the actual target. Snapshots of the output from the generic detector can be seen below in Figure 4.4.

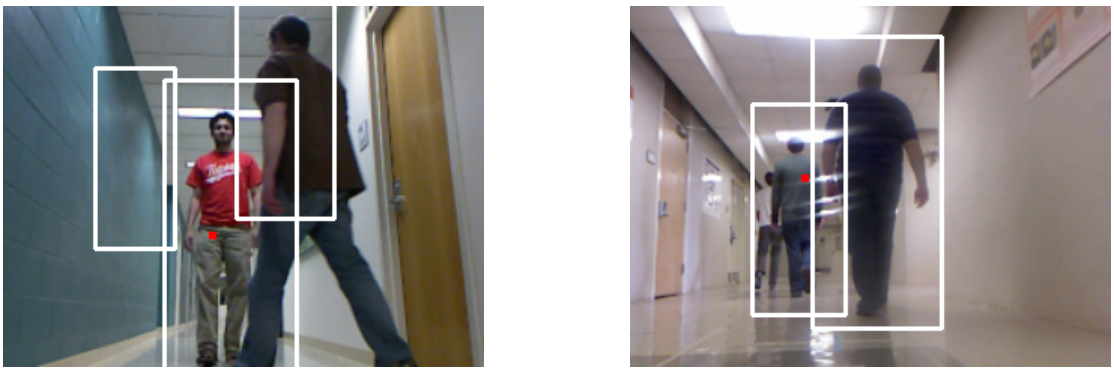

Figure 4.4: Generic HOG detections

4.4 Specific Appearance Model

The specific appearance model is the working piece of the algorithm that makes distinctions between people possible. It helps create distinction between the generic detections and provides a method for detecting occlusions.

It is used in the particle filter weight update to scale the state observations. If the algorithm were to rely solely on the hybrid state locations of each particle for specific target detection, we would find that the particles would often drift to other non-target detections. If the particle filter was not scaled by this appearance factor and relied solely on the spatial information, particles would gain weight when they became close in position to any person detection and eventually attach themselves to that detection regardless of their appearance similarity to the target person. This would result in the eventual loss of the target person. The algorithm would then assume the new person it is tracking is the target person.

It is also used as a final check on the expected target position to determine if an occlusion has occurred. The particle filter implementation, regardless of the scaling effect influence the specific appearance model has on the expected state of the target, still gives an expected target position at every iteration. This means that there will

always be an expected target state coming from the particle filter and we have no way of knowing for sure if the expected target state is the correct person or not. It is possible that this is just another non-target person occluding our target person. Therefore, the specific appearance model is used as a check against the expected target state to make a final decision as to whether or not this is the correct target.

The specific appearance model is created with a color histogram. The color histogram used in this algorithm employs 10 bins on each color channel. Each color channel ranges in value from 0 to 255. This means that on each color channel a bin has approximately 25.6 values assigned to it. This is of course not exact because a color channel value is not a float. With 10 bins per color channel, the entire color histogram has 1000 possible entries. Refer to section 3.5 for color histogram theory. The purpose of this color histogram is to make a model of some window area for the purpose of later comparison.

The initial color histogram is the color histogram that is used for the remainder of the test period when considering possible person detections as the target detection. It used essentially as a template. To create the initial color histogram the depth information from the RGB-D sensor is first used. When the user initially clicks the image to signify the target person, an average depth is calculated around this clicked area. Then pixels inside the initial selection window that lie outside a given threshold of this average depth are not considered while creating the color histogram. The threshold used for this determining whether or not a pixel is close enough to the average is 0.3 meters. In Figure 4.5 you can see examples of the pixels used to construct the color histogram. Notice that the black areas correspond to pixels that lie outside of the average depth area. Once the initial color histogram has been constructed, no additional models are constructed.

Figure 4.5: Determining pixels to be used in color histogram

4.4.1 Appearance Model Influence on Weight Updates

During the particle weight update, the initial specific appearance model (or color histogram) is used to scale the spatial informations influence on the weight. This is done by taking the particle state of the particle currently being updated and creating a color histogram for the bounding box that will represent this particle. The same method is used as was in creating the initial appearance model. Meaning, the average depth is found and any pixels that lie outside the average depth value plus the threshold are not considered when making the color histogram. The color

histogram of the particle state under consideration is then compared with the initial color histogram using the L2 norm. This then gives the the value Δ_{d_s} from equation 4.2 which is used as the scaling factor on the weight update of the current particle in equation 4.4.

4.4.2 Appearance Model for Occlusion Detection

As a final system check, the specific appearance model is used to determine whether or not the expected target state is the correct target or if an occlusion has occurred. Remember that the output from the particle filter is an expected state and this expected state is taken to be the expected position of the target. However, as was stated before, the particle filter will always give us an expected target state and there is no way of knowing if it is the correct target or if the target has been occluded. To remedy this problem, we use the specific appearance model as the final check. The expected target state is taken to this stage of the algorithm. Below in Figure 4.6 you can see the template for the initial histogram of a scene and in Figure 4.7 the templates for the color histograms that will be compared against the initial color histogram can be seen.

There is an additional element that comes into play at this point in the algorithm. Since the expected position has no bounding box associated with it and there is no way for the algorithm to currently know the actual size of the bounding box that would surround the real target, so an average of all detection bounding boxes is used as the theoretical bounding box. This is the bounding box that is used to create the color histogram centered around the expected target position. This color histogram once again is created as before. The average depth is found and any pixels lying outside the average depth within the threshold are not considered when making

Figure 4.6: Initial template for the color histogram

Figure 4.7: Expected position color histogram templates from frames in the sequence

the color histogram. Once the color histogram has been created for the expected target position, it is compared against the initial specific appearance model (color histogram) using the L2 norm. If the resulting difference is below the threshold, t_a, then the expected target person is assumed to be the correct target person. If the resulting difference is not below the threshold, t_a, then an occlusion is declared. The threshold value t_a is varied between 0.15 and 0.35 in execution of this algorithm. The condition that must be satisfied can be seen below in Equation 4.7. It should be noted here that if an occlusion is declared, then a missed detection is added to the

counter that is used by the motion model.

$$\|d_s^{expected} - d_s^{orig}\| < t_a \tag{4.7}$$

4.4.3 Appearance Model Occlusion Check Pseudocode

1. Create color histogram at the expected state
2. Compare this to the initial color histogram (Equation 4.7)
3. If the result is less than t_a
    - Target found
4. else
    - Target occluded

4.5 Robot Control Architecture

The input to the robot control architecture is the targets last position, the targets current position, and an occlusion flag. The overall procedure of the control architecture can be seen below in Figure 4.8.

Once the control architecture has the previous and current target positions, there are two parameters that will instruct the robot to take no movement action and stop. The first parameter is the occlusion flag. If it has been set, then the robot executes a stop command. The robot does not update the last position and returns to the tracking algorithm. However, if the the occlusion flag has not been set then the second parameter considered is the magnitude of the movement. Since it is unlikely that from frame to frame the target will move a very large distance, a check is set

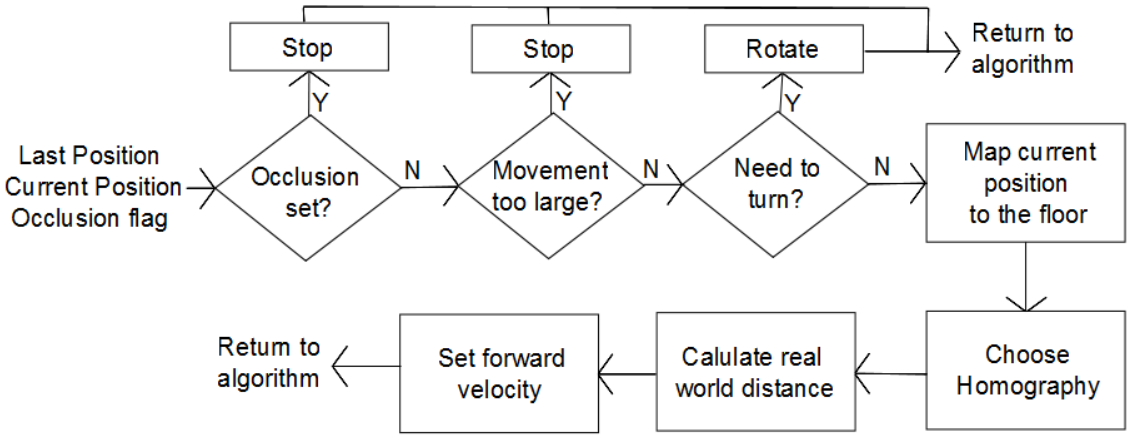

Figure 4.8: Robot control architecture

in place to identify a large distance movement. The purpose for this check is to be careful in the event that the tracker somehow misidentifies the target. We do not want the robot to try to make a large movement that will set it off course for further tracking. Therefore, if it is determined that the distance traveled from the previous target detection is too great the robot will execute a stop command. Once again it will not update the last position and will return to the tracking algorithm.

If neither of these flags have been set then the robot control architecture will first determine if a turn is necessary. The need to turn is checked first for three reasons. First, since testing is being done in a hallway, it is very rare that the robot will ever have to turn. Second, we want to keep the target centrally located in the image in order for the particle filter motion model to function properly. And third, if a rotation is required, we do not want to set a rotate and forward movement command at the same time because whichever command is given first will not be executed. The check for a turn command is fairly simple. The control architecture checks to make sure the center of the target is within the middle three fifths of the image. If the target is outside of the middle three fifths then depending on whether the target is

in the right third or the left third the robot executes a rotate right or rotate left command. Once this command has been set, the robot control architecture sets the current target position to the previous target position and returns to the tracking algorithm.

If a turn is not necessary then the robot evaluates the velocity at which it needs to move forward or backward. This is done in a few steps. First the control architecture maps the current position to the ground plane. Then, depending on which section of the ground plane this mapped point is in the control architecture chooses a homography. The ground plane is divided into three separate sections. The reason for this is because the pixel distance between two pixels further off in the ground plane correlate to a larger real world distance than the real world distance that correlates to the pixel distance between two pixels that are closer in the ground plane.

Three separate transformation homographies were calculated off-line. This off-line calculation was done by taking pictures (at the same height as the camera is on the robot) of a piece of cardboard with known lengths. Then the images were brought into a program that allows the user to click on the corners of the piece of cardboard. Corresponding the distances between the clicked points and the known real world lengths allows for the construction of a homography, as discussed in section 3.2, that will allow us to transform points from image space to real world space.

Once the correct homography has been selected, the real world distance between the previous position and the current position is calculated and the robot forward velocity is set such that it will move this distance. After the set forward velocity command has been set, the robot control architecture sets the current target position to the previous target position and returns control over to the tracking algorithm.

4.5.1 Robot Control Pseudocode

1. Input: last position, current position, occlusion flag
2. If occlusion flag set (Section 4.4.2)
    (a) Stop robot
    (b) Exit robot control
3. Else if movement from last position to current position is too large
    (a) Stop robot
    (b) Exit robot control
4. Else if robot needs to turn
    (a) Turn robot in appropriate direction
    (b) Set last position to current position
    (c) Exit robot control
5. Else
    (a) Map current position to ground plane
    (b) Choose homography based on which third of the ground plane the current position is in
    (c) Calculate the real world distance between the current position and last position
    (d) Set robot forward velocity
    (e) Set last position to current position
    (f) Exit robot control

Chapter 5

Experiments and Results

The algorithm was run on multiple test sets to determine how well it would perform in different situations. The there were two test environments both of which were located indoors in areas of minimal traffic. The setup used is the Microsoft Kinect camera, the Pioneer P3-AT robot, on a Dell Studio 18 laptop with an i7 core processor. The operating system being used is Ubuntu 11.04. ROS is used to connect to and retrieve information from the camera and the ARIA library is used to control the mobile robot.

We want to evaluate the algorithm performance in three main areas. The first is scenes with no occlusion of the target, the second is scenes with minimal occlusion and the third is scenes with large occlusions. The purpose for the three separate test areas is to test the needs of person following algorithm which is threefold. (1) It needs to be established that the baseline system works, (2) it needs to be established that the algorithm can handle brief occlusions, and finally (3) the system should be able to recover the target after a long term occlusion.

5.1 Types of Images

There are several different types of images that will be seen in the following sections. To avoid the need to explain what it is the images actually represent, this section will serve to show one type of each image for a single frame. The six image types can be seen below in Figure 5.1

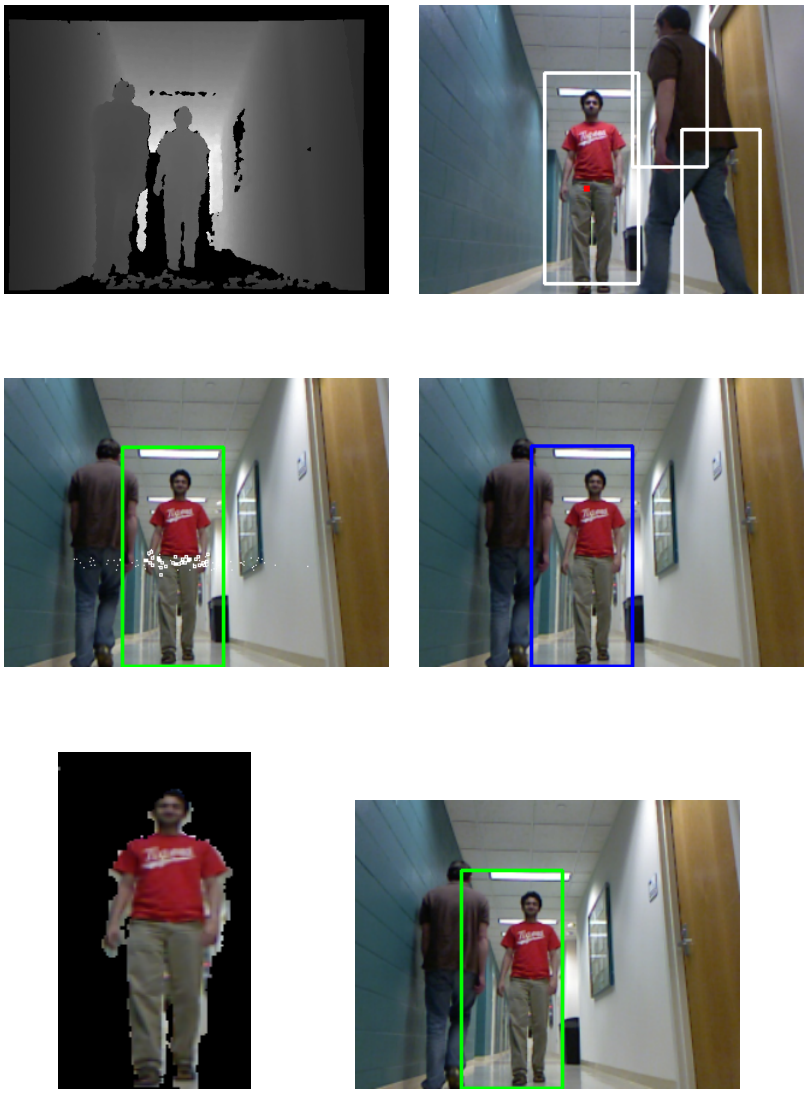

Figure 5.1: Types of images

Each image is explained in order from top left to bottom right. The first

image is the depth image. It is a representation of the data that is provided from the depth sensor. The second image shows every HOG person detection provided from the generic person detector. The third image is the image with the final target detection bounding box and the particle positions overlayed on top of it. The fourth image shows the particle filter expected position bounding box. The fifth image is a representation of the template that will be used to create a color histogram. And the sixth image is simply the current image with the positive target detection bounding box lain over it.

5.2 No Occlusion

The test sets for non-occlusion cases will consist of simple test cases in which the target person is walking down the hallway in a straight path. There will be no full or partial occlusions. There will however be other people present in the scenes. The purpose of these tests is to evaluate the system response to baseline test cases. In this section we want to determine the system response to different cases of non-occluding robot/target interaction. To begin, in Figure 5.2 the initial progression of a single target down the hall can be seen with the corresponding depth images.

Figure 5.2: No occlusion and different depths

In this initial run, the target individual is walking down the hall with only

one additional person present. The additional person is at a different depth. The system performs as expected in this case and is easily able to follow the target. Next, we want to test the system when another person appears at approximately the same depth and relative location as the target. This case can be seen below in Figure 5.3.

Figure 5.3: No occlusion and similar depths

It can be seen from the segment in Figure 5.3 that when an additional person within proximity of the target appears in the scene with a similar depth, the tracker stays with the target person and is not influenced by the sudden appearance of another person. This is due to the relatively small particle propagation that is allowed whenever a positive target identification has occurred.

For the final non-occlusion test, the system is tested for a target change in appearance. Only the initial template is used to create the color histogram, thus, the integrity of this method must be tested for cases in which the target changes pose. Below in Figure 5.4 the initial template for the image sequence can be seen and in figure 5.5 the detection images for later frames of the sequence are displayed.

It can be seen that the system is indeed robust to appearance changes in the target. Regardless of the initial color histogram template that results from the target facing away from the camera, a side or forward pose of the target is not enough of an appearance change to set off an occlusion flag in the appearance comparison stage of the algorithm. It should also be noted that the appearance model is unaffected by

Figure 5.4: Initial color histogram template

Figure 5.5: No occlusion with different poses

the illumination from the ceiling lights. The illumination changes the color histogram template by creating streaks across the image, however, the appearance model is still robust enough to not consider this change as an occlusion.

5.3 Brief Occlusions

The previous section tested the systems robustness to the point that the target was able to be tracked in three situations. The first being a scene with multiple people in the image at different depths, the second being multiple people in the scene at

similar depths and the third being appearance changes of the target. Now we would like to test the system against brief occlusions.

In order to test the system against brief occlusions a scenario is set up such that the target person is fully occluded for a brief period of time while walking down the hall. To show the effectiveness of the occlusion detection, the detection images are compared against the particle filter expected position images, depth images, and the color histogram templates. Below in Figure 5.6 and 5.7 two separate occlusion instances, and the systems response to these instances, can be seen. The first occlusion instance in Figure 5.6 compares the detections against the particle filter expected positions and the second occlusion instance in Figure 5.7 compares the detections against the depth images and the color histogram templates (where the first image is the initial template used to create the color histogram).

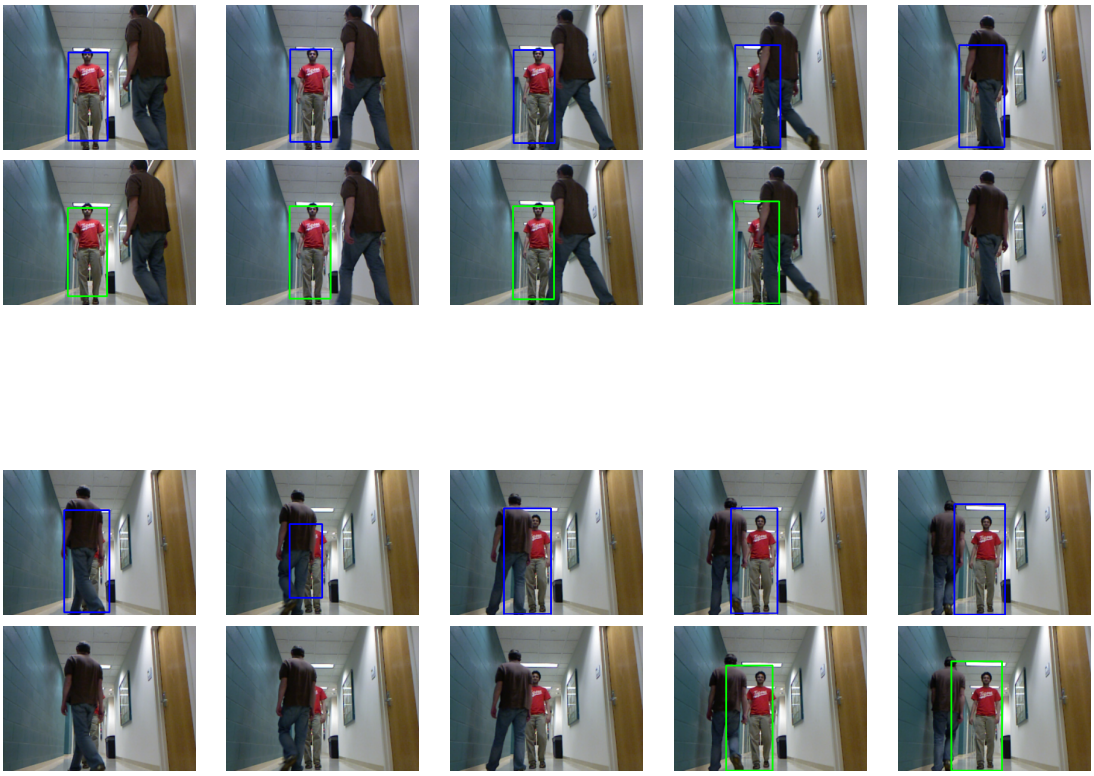

Figure 5.6: Brief occlusion instance one: particle filter expected positions

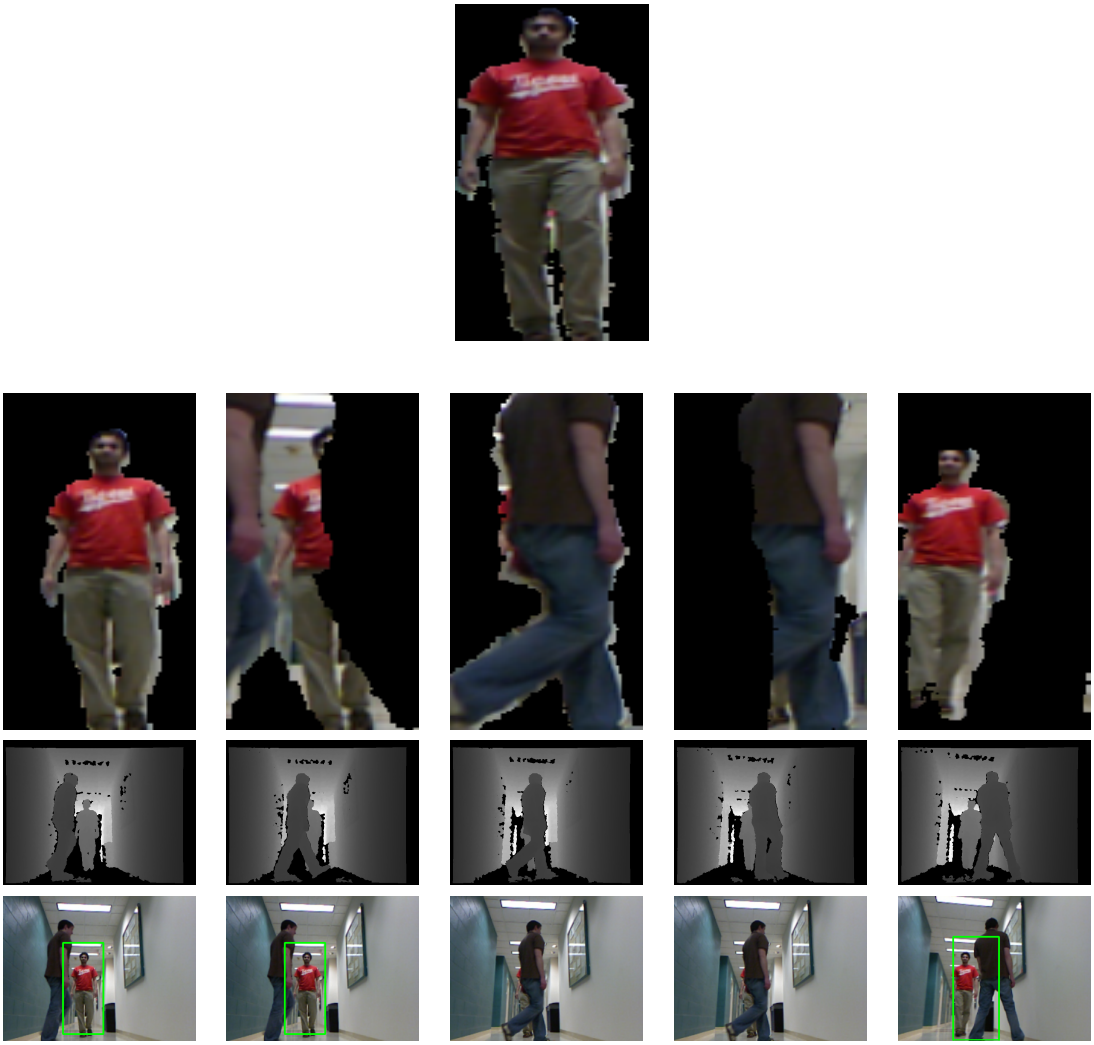

Figure 5.7: Brief occlusion instance two: depth images and color histogram templates

From these two instances, it can be seen that the system is able to handle brief occlusions. The particle filter expected state exists in the area where the occluding person is, however, the appearance model on the tail end of the system registers an appearance change signifying the detected person is an occluding person and should not be confirmed as the target person. The threshold value t_a used for this image sequence is a value of 0.15. The appearance difference is above the threshold, so the system declares that an occlusion has occurred and overrides the particle filter

expected position. Were the appearance model not present, the occluding party would be chosen as the target person and quite possibly end up being the new target person.

5.4 Long Occlusions

Since the testing for brief occlusions was successful, the final system test is the response to a long term occlusion. It needs to be established that the system can reacquire the target after it has lost the target for more than three frames (which is the standard in the brief occlusion). To conduct this test, the algorithm is run on a sequence in which the target is occluded for ten frames before reappearing. The detection results of this test can be seen in Figure 5.8.

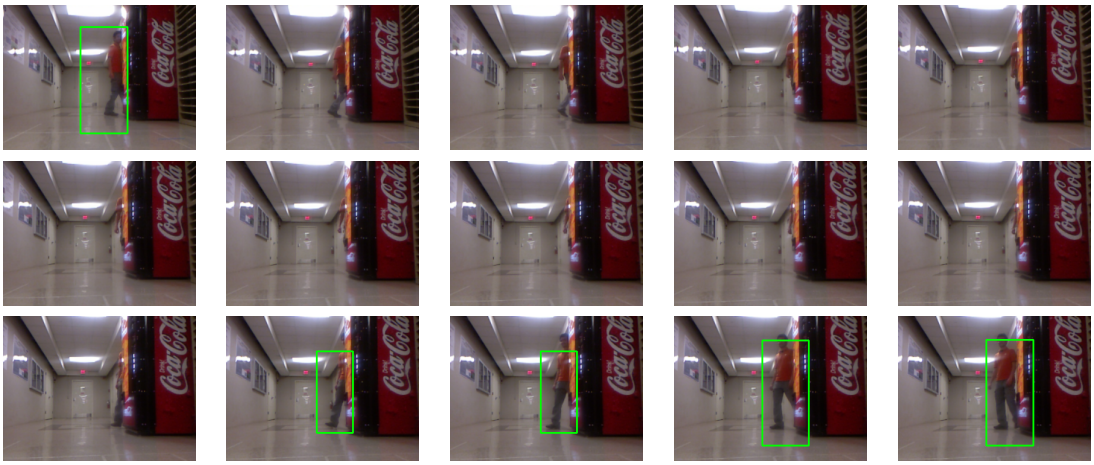

Figure 5.8: Long time occlusion: 10 frame occlusion

This sequence shows that the system is able to recover from a long term occlusion. The reason that this is possible is due to the fact that the appearance model does not change after the initial frame. The low magnitude of particle propagation also makes this possible. The systems ability to recover from a long term occlusion is a positive aspect, however, a trade-off is made in order for the system to obtain this quality. The trade-off comes in tracking target lateral motion and will be further

discussed in the next chapter.

5.5 Overall Performance

As a means of determining the systems overall performance, the intersection over union of the detected target position versus the true target position was plotted for the sequence during which brief occlusions occurred. The occluded frames were not included in this graph. The intersection over union is calculated by dividing the area in which the detected position and true position intersect by the total area of the two. The graphed sequence can be seen below in Figure 5.9.

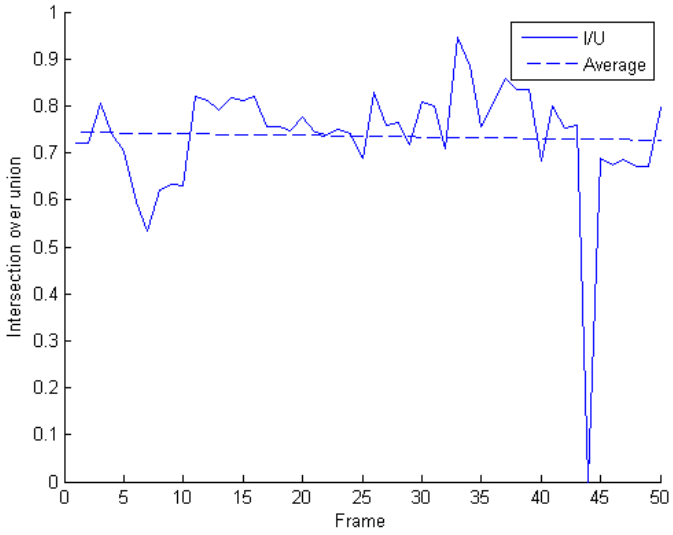

Figure 5.9: Overall performance

The graphed data shows us that the system performs quite well overall. It maintains about a 75 percent intersection over union throughout the sequence. It should be noted that not every frame from the sequence is graphed. For a total frame breakdown refer to Table 5.1.

True positives	79
True negatives	9
False positives	0
False negatives	12

Table 5.1: Frame breakdown

From this table it can be inferred that the system, though detecting every occlusion, does make a detection trade off. There were 12 frames during which the target was not found even though visible in the scene. The reasons for this will be discussed in the conclusion section.

Chapter 6

Conclusions

This project involved developing an algorithm to track and follow a person through a variety of situations. Overall, the algorithm performed well. It has been demonstrated that the system is able to track a target person in the three situations that were initially proposed. The algorithm can track an individual that is not being occluded when environmental clutter and other people are present, the algorithm can track an individual through brief occlusions, and it can reacquire a target individual after a long time occlusion.

That being said, the trade offs to achieve this performance must be discussed. The biggest trade off comes in the particle motion. In order to reacquire the target after a long time occlusion the system must be set up such that the particles do not scatter too much at each iteration. If the particles stay centrally located and only slightly drift at each iteration then a central location can be maintained. Maintaining a central location is key in reacquiring the target individual after a long time occlusion. Since the particles do not make drastic propagations at each iteration, the bulk of them will stay in a reasonably close proximity to the last target detection. Thus, when the target disappears for a long time and then reappears the particles have not

scattered about the image making it impossible to reacquire the target. This method has been shown to work, but it does make the system susceptible to false negatives. These false negatives occur when the target moves rapidly from one side of the image to the other. Since the magnitude of the particle propagation is small, the particles will not be able to keep up with the rapid movement. The target expected state is then in a location in which no person, or the wrong person, is present and the appearance model classifies the expected position as an occluded target.

The algorithm is also susceptible to large changes in scale of the target. If the target starts off large and through the sequence progression is reduced significantly in size, the system will no longer be able to detect the target. This stems from the fact that the window size being used to estimate the size of the expected position is an average of all windows. Though the generic HOG detector does search at multiple scales, there is a limit to the scale size that is most likely forced upon the detector by the training images that were used. This means humans that are off in the distance do not get detected. For this algorithm that translates to the actual target bounding box size not being considered in the average of all detection boxes. This means the expected position bounding box will be much larger than it should be and will not provide a good template for the color histogram. The appearance model will then classify the expected position as an occlusion and a false negative will occur.

Other than these two issues, the system performed well. It is a real time system which is a necessity for the following portion of the algorithm. In the next section some possible system improvements will be discussed.

6.1 Future Work

Areas of future work should aim to improve on the two major deficiencies of the algorithm. These two deficiencies are the particle motion model and scale issues. The latter could most likely be solved by training a HOG detector that considers smaller sized targets as positive human detections, while the former would entail creating a more complicated motion model for the particle filter. It is possible that the motion problem could be solved by allowing the magnitude of particle propagation to increase but add more particles, however this will slow the system down considerably. Since it is necessary for the algorithm to run in real time, increased processing time at each frame is undesirable.

One other possible improvement that could be made to the system would be incorporate a target motion history. This could be implemented by keeping a history of target positions in a top down coordinate system. This target history could then be used to calculate the target trajectory which could be used as another occlusion check. In this theoretical implementation, both the appearance model and the trajectory could be taken into consideration when making a decision about a possible occlusion.

One final addition that could be contributed to the algorithm is the concept of creating an adaptive appearance model. To implement this, we would not only keep the initial appearance model around for comparison but also keep a the previous 4 or 5 appearance models around. It is possible that this could help with the scale issue. If the target is getting smaller it would be also be smaller in the updated appearance models giving the algorithm more of a chance to classify the smaller target as non-occluded.

Bibliography

[1] Mayank Bansal, Sang-Hack Jung, Bogdan Matei, Jayan Eledath, and Harpreet S. Sawhney. A real-time pedestrian detection system based on structure and appearance classification. In *ICRA'10*, pages 903–909, 2010.

[2] D. Beymer and K. Konolige. Tracking people from a mobile platform. In *Proceedings of the International Joint Conference on Artificial Intelligence.*, 2001.

[3] Jonathan Brookshire. Person following using histograms of oriented gradients. *International Journal of Social Robotics*, 2:137–146, 2010.

[4] Zhichao Chen and Stanley Birchfield. Person following with a mobile robot using binocular feature-based tracking. In *International Conference on Intelligent Robots and Systems*, 2007.

[5] G. Chivilo, F. Mezzaro, A. Sgorbissa, and R. Zaccaria. Follow-the-leader behaviour through optical flow minimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, volume 4, pages 3182–3187, 2004.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.

[7] T. Germa, F. Lerasle, N. Ouadah, and V. Cadenat. Vision and rfid data fusion for tracking people in crowds by a mobile robot. *Computer Vision and Image Understanding*, 114(6):641 – 651, 2010. Special Issue on Multi-Camera and Multi-Modal Sensor Fusion.

[8] H. Grabner and H. Bischof. On-line boosting and vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 260 – 267, June 2006.

[9] Peter Henry, Christian Vollmer, Brian Ferris, and Dieter Fox. Learning to navigate through crowded environments. In *International Conference on Robotics and Automation*, 2010.

[10] Frank Hoeller, Dirk Schulz, Mark Moors, and Frank E. Schneider. Accompanying persons with a mobile robot using motion prediction and probabilistic roadmaps. In *IROS*, pages 1260–1265, 2007.

[11] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[12] Thorsten Joachims. Advances in kernel methods. chapter Making large-scale support vector machine learning practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[13] Rachel Kirby, Reid Simmons, and Jodi Forlizzi. Variable sized grid cells for rapid replanning in dynamic environments. In *Proceedings of the IEEE/RSJ international conference on Intelligent robots and systems*, pages 4913–4918, Piscataway, NJ, USA, 2009. IEEE Press.

[14] Hyukseong Kwon, Youngrock Yoon, Jae Byung Park, and Avinash C. Kak. Person tracking with a mobile robot using two uncalibrated independently moving cameras. In *IEEE International Conference on Robotics and Automation*, pages 2888–2894, 2005.

[15] Matthias Luber, Luciano Spinello, and Kai O. Arras. People tracking in RGB-D data with online-boosted target models. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, USA, 2011.

[16] M. Piaggio, R. Fornaro, A. Piombo, L. Sanna, and R. Zaccaria. An optical-flow person following behaviour. In *Proceedings of the IEEE ISIC/CIRA/ISAS joint conference.*, pages 301–306, 1998.

[17] Donald B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24:843–854, 1979.

[18] Christian Schlegel, Jrg Illmann, Heiko Jaberg, Matthias Schuster, and Robert Worz. Vision based person tracking with a mobile robot. In *In Proceedings of the British Machine Vision Conference*, pages 418–427, 1998.

[19] William Robson Schwartz, Raghuraman Gopalan, Rama Chellappa, and Larry S. Davis. Robust human detection under occlusion by integrating face and person detectors. In *Proceedings of the Third International Conference on Advances in Biometrics*, pages 970–979. Springer-Verlag, 2009.

[20] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[21] H. Sidenbladh, D. Kragic, and H.I. Christensen. A person following behaviour for a mobile robot. In *IEEE International Conference on Robotics and Automation.*, volume 1, pages 670–675, 1999.

[22] Luciano Spinello and Kai O. Arras. People detection in RGB-D data. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[23] M. Tarokh and P. Ferrari. Robotic person following using fuzzy control and image segmentation. *Journal of Robotic Systems*, 20(9), 2003.

[24] C. Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, April 1991.

[25] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. In *ACM SIGGRAPH 2006*, pages 1160–1168, 2006.

[26] P.A. Vela, A. Betser, J. Malcolm, and A. Tannenbaum. Vision-based range regulation of a leader-follower formation. *IEEE Transactions on Control Systems Technology*, 17(2):442–448, 2009.

[27] Paul Viola and Michael Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[28] Xiaoyu Wang, Tony X. Han, and Shuicheng Yan. An HOG-LBP human detector with partial occlusion handling. In *IEEE 12th International Conference on Computer Vision*, pages 32 –39, 2009.

[29] Qiang Zhu, Shai Avidan, Mei-chen Yeh, and Kwang-ting Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, pages 1491–1498, 2006.