Person Tracking From a Dynamic Balancing Platform

Dinesh Govindaraju, Brett Browning, Manuela Veloso
November 2004
CMU-CS-04-181

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213

This research was sponsored by the United States Army under Grant No. DABT63-99-1-0013. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of DARPA, the US Army, or the US Government.

Keywords: vision, person tracking, mean shift

Abstract

Recently, we have begun investigating a new robot soccer domain built around the concept of human-robot teams in a ‘peer’ setting. One of the key challenges for addressing effective human-robot interaction is to robustly identify and track people and robot teammates without requiring undue prior knowledge of their appearance. For cost and complexity reasons, our robots are equipped with monocular color cameras. Thus, we seek an algorithm to enable reliable acquisition and tracking of people and robots from a robot armed with a monocular color camera. We have developed a novel algorithm for acquiring and tracking a single human subject from a dynamically balancing platform, a Segway RMP robot, using a monocular color camera. Our technique uses a combination of known vision and tracking techniques including region growing, motion detection, and mean-shift color-template tracking. In this paper, we describe our approach, and analyze its performance and limitations, for both acquiring and tracking a single human target in an indoor environment. Our experiments demonstrate that acquisition and tracking are feasible with a monocular camera even for a dynamically balancing platform. Moreover, our results show that with current processor technology real-time tracking and robot response are achievable.

1 INTRODUCTION

Recently, we have begun developing a new domain, called Segway Soccer, for studying human-robot interaction within the confines of an adversarial task [8]. This domain is in many ways an extension of existing RoboCup domains [7] to incorporate human-robot teams in a 'peer' partnership. For such human-robot interaction to be realizable in its full scale, each robot will need the ability to reliably identify and track its human and robot counterparts, preferably without requiring in-depth prior knowledge of their appearance.

There have been a number of investigations into the problem of tracking people using static cameras, in both indoor and outdoor environments (e.g. [1]). Within the robotics community, there has been a similar interest in tracking people from robot platforms using either laser range finders or combinations of range finders with and vision sensors [6]. To our knowledge, there is as yet no work that focuses on using color-based monocular vision as the *only* sensing modality. Monocular color cameras are appealing due to their low cost and complexity, wide availability and ever improving performance driven by the consumer market. These issues are important when building teams of robots, as reducing cost and complexity is of paramount importance. Therefore, we seek a technique that enables reliable *acquisition* and *tracking* of human subjects by a robot using an affordable monocular color camera without undue prior knowledge of their appearance.

We believe that when using a novel combination of standard vision techniques found in both surveillance and robotics literature, this task is achievable. In this paper, we describe our efforts to develop such an approach using the unique, dynamically balancing Segway RMP platform (see Fig. 1.) and analyze the performance of our resulting implementation on the Segway RMP platform.

Figure 1. The Segway RMP robot with its primary means of perception, a monocular color camera on a custom pan-tilt unit.

The paper is structured as follows. In the next section, we describe the focus problem of this paper: tracking human subjects from the Segway RMP platform using a monocular color camera. In section 3, we describe our approach to the problem and the techniques we have developed. In section 4 we present experimental results detailing the performance of our approach. In section 5, we describe the relevant related work in the literature leading to our conclusions in section 6.

2 PROBLEM DESCRIPTION

In this paper we focus on the problem of enabling a robot, in this case a Segway RMP platform, to detect a person and then track the person as he or she moves about the environment. Once able to track a person, a number of human-robot interaction tasks become feasible. For a general example, given the Segway's payload capability following the person while carrying a load becomes feasible. Another example would be to train the robot by following a person performing the desired actions and subsequently imitating them (e.g. [10]).

The Segway RMP is a commercially available robot platform[1] based on the Segway HT mobility platform, a two-wheeled self-balancing scooter. It comes equipped with micro-controllers and rate gyros that enable it to self-balance. In our prior work, we have equipped the Segway robot with two mid-end laptops and single monocular camera mounted on a custom pan-tilt unit at approximately 1.4m from the ground [11]. All processing must take place on-board the robot. One laptop is dedicated to motion control and forms the interface to the Segway RMP base. The second laptop is dedicated to vision, behaviors, logging, and navigation. For the purposes of this discussion, the vision laptop is an Intel Pentium III operating at 1.7GHz. In order to get reasonable responsiveness from the robot to dynamic changes in the environment, the robot control loop must operate as fast as possible and with minimum latency. As there are unavoidable delays induced by the image capture process, it becomes imperative that image processing and the resulting behavior generation operate within the cycle of a single frame capture. Thus, all image processing algorithms must be very efficient. Finally, the self-balancing nature of the robot induces small perturbations in the pitch angle of a few degrees, which results in significant apparent up-and-down motion in the imaging plane dependent upon the pan-tilt angles of the camera. The camera is a low-cost web-camera, in particular the Logitech Pro 4000, providing images via USB at 30Hz with 320x240 pixels in a YUV 4:2:0 planar format.

Figure 2. An example camera view from the Segway RMP.

We wish to limit the required prior knowledge of the appearance of the human subject and knowledge of the lighting conditions in order to improve the general applicability of our algorithm. Additionally, we wish for the system to be completely autonomous in order to achieve running on multiple robots without requiring a large human-operator team. We now describe our developed technique to address this problem.

3 Selected Approach

Within the vision and robotics literature, there are a number of techniques that are applicable to this problem. One approach is to rely on face detection to be able to identify where a human subject is, and to then track that face over time. Such an approach has found use in platforms such as the Sony Qrio and AIBO for 'face-to-face' human-robot interaction, but is of limited use for tasks such as following a human subject where their face may not always be visible. Given the generality of our task, such a technique is not useful here.

In contrast, there have been a number of tracking techniques developed within the vision surveillance community. Most notably, techniques based on the mean-shift algorithm (e.g. [2]) have proven successful at tracking through lighting and scale changes in real-world scenes. Moreover, for a small number of tracked objects it is both robust and efficient.

In vision surveillance, the mean-shift algorithm is often used for tracking colored patches over time. Thus, it is only useful for tracking a target once identified. That is, a challenge is how to first *acquire* the target to be tracked. For a robot performing autonomous tracking, there is the additional problem of behavior. How should the robot move in order to acquire a subject and once

[1] *http://www.segway.com/rmp*

acquired, how should it follow the subject? Finally, how should it switch between these different modes of operation? To maintain the robust overall performance, we address this problem by using a state machine approach to divide between the two distinct modes of operation, namely acquisition and tracking. Such an approach builds on our prior work using the Skills, Tactics, Plays (STP) architecture to control a single robot using behaviors constructed in a finite state machine arrangement with transitions controlled by internal state or perception of the world [9]. Fig. 3 shows the state machine we developed which begins in the acquisition behavior and executes until a suitable object to track is found. This Acquisition State itself consists of both Motion Detection and Potential Buildup. Once a suitable object is found, the robot tracks it until it is lost. Thereafter it returns again to the acquisition behavior until another target is acquired.

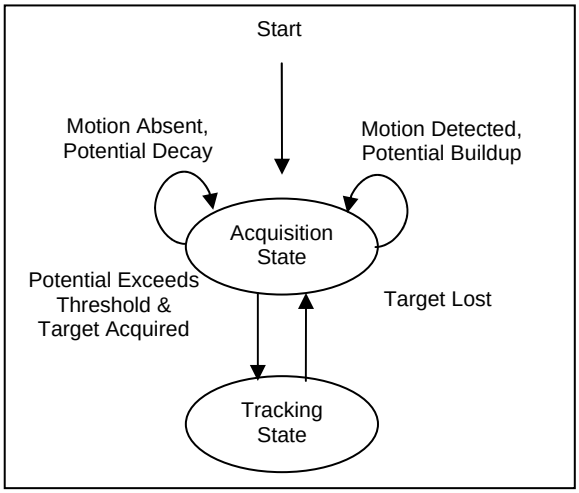

Figure 3. Person Tracking Behavior Finite State Machine.

We now describe the operation of the acquisition and tracking in greater detail.

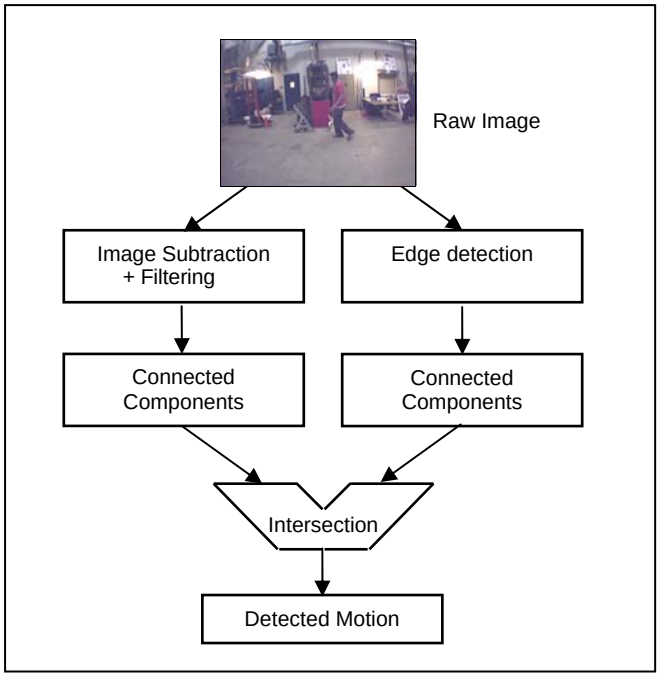

Figure 4. Process of Motion Detection

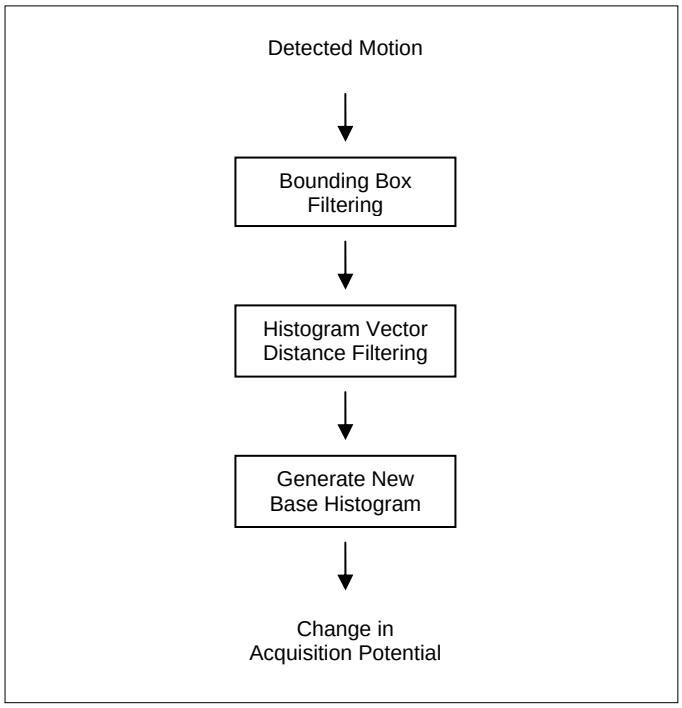

Figure 5. Process of Potential Build Up

If there is no applicable motion detected in the frame, the acquisition potential decays exponentially. When this potential increases and exceeds a certain threshold value, the system acquires the target and transitions into the tracking state where tracking commences using the mean-shift algorithm [1]. The acquisition potential exists to address the very real concern that the robot, when in an acquisition state, encounters a person who randomly walks in front of it and erroneously targets that person to track. As such, the acquisition potential ensures (with adjustment of the threshold potential) that only a person who desires to be tracked is targeted.

3.1 Target Acquisition

We will now proceed to outline the various operations that target acquisition consists of.

3.1.1 Image Subtraction with Offset Filter

The acquisition phase begins with image subtraction where motion in the image is detected by comparing each pixel from the current frame with the preceding frame. The difference in the color space values of the pixel from the two frames is calculated, and if this difference is greater than some threshold value, the pixel is marked as having moved [12]. One aspect to note about this procedure is that when running image subtraction on the Segway RMP as it is balancing, many undesired pixels are marked as the rocking motion caused by balancing leads to unwanted vertical motion of pixels. This is overcome by applying an offset filter which is combined with image subtraction as in Table I , to ensure that any motion that is detected cannot be purely vertical in nature, but must horizontal as well.

```
Image_Subtraction(){
   // pix_t(x,y) : pixel at position x,y on image frame at time t
   // moved : array to indicate pixel motion
   for x,y ∈ image
     if pix_t(x,y)== pix_t-1(x,y) || pix_t(x,y)== pix_t-1(x,y-1) || pix_t(x,y)== pix_t-1(x,y+1)
       moved[pix_t(x,y)] = false
     else
       moved[pix_t(x,y)] = true
     endif
   endfor
}
```

TABLE I. IMPLEMENTATION OF IMAGE SUBTRACTION AND OFFSET FILTER

Improvement in results when applying this offset filter can be seen in Fig. 7 & 8.

3.1.2 Morphological Filtering and Region Growing

Once pixels of interest have been marked in the target acquisition phase through the manner above, there still tends to be a high proportion of noise. This is overcome by running a variant of morphological filtering as given in Table II, which acts to remove marked pixels which themselves are not surrounded by a threshold number of other marked pixels. This is due to the fact that the triggering motion[2], marks pixels in patches leaving single stray pixels to indicate noise.

```
Morphological_Filtering(){
   // pix_t(x,y) : pixel at position x,y on image frame at time t
   // moved : array to indicate pixel motion

   for x,y where moved[pix_t(x,y)] == true
     for x' where |x'-x| < 3
       for y' where |y'-y| < 3
         if moved[pix_t(x',y')] != true
           nonmarked++
         endif
       endfor
     endfor

     if nonmarked > threshold
       moved[pix_t(x,y)] = false
     endif
   endfor
}
```

TABLE II. IMPLEMENTATION OF MORPHOLOGICAL FILTERING

Results of morphological filtering can be seen in Fig. 8, and after this stage, remaining marked pixels are likely to be those that indicate boundaries of motion. Region growing is then carried out which involves examining both vertical and horizontal lines in the image, and if two marked pixels in an examined line are separated by less than some given threshold distance, pixels in between the marked pixels are marked as well as given in Table III.

[2] In our case the triggering motion is of the person to be tracked rocking his body slowly from side to side to maximize horizontal motion

```
Region_Growing(){
   // pix_t(x,y) : pixel at position x,y on image frame at time t
   // moved : array to indicate pixel motion

   for 1 < x < image_width
     if moved[pix_t(x,y_1)] == true && moved[pix_t(x,y_2)] == true && (y_2 - y_1) < threshold
       for y_1 < y_between < y_2
         moved[pix_t(x,y_between)] = true
       endfor
     endif
   endfor

   for 1 < y < image_height
     if moved[pix_t(x_1,y)] == true && moved[pix_t(x_2,y)] == true && (x_2 - x_1) < threshold
       for x_1 < x_between < x_2
         moved[pix_t(x_between,y)] = true
       endfor
     endif
   endfor

}
```

TABLE III. IMPLEMENTATION OF REGION GROWING

This has the effect of indicating entire pixel regions that are believed to be moving and results of this procedure can be seen in Fig 9.

3.1.3 Edge Detection

At this stage, edge detection is carried out by applying Sobel Edge Detection [3] to the original raw image, producing outlines of the various regions in the image. This procedure involves first computing the 3 by 3 convolution kernels as shown in Fig. 6.

-1	0	+1
-2	0	+2
-1	0	+1

+1	+2	+1
0	0	0
-1	-2	-1

Figure 6. Sobel Convolution Masks G_x (left) and G_y (right)

The approximate gradient magnitude for each pixel is then computed as given in (2).

$$|G| = |G_X| + |G_Y| \qquad (2)$$

Pixels are then marked if their gradient magnitude is found to be above some threshold value as in Table IV.

```
Edge_Detection(){
   // pix_t(x,y) : pixel at position x,y on image frame at time t
   // grad(x,y) : gradient magnitude of pixel at position x,y on image frame at time t
   // outline : array to indicate if pixel is an outline

   for x,y ∈ image
     if |grad_t(x,y)| > threshold
       outline[pix_t(x,y)] = true
     endif
   endfor
}
```

TABLE IV. IMPLEMENTATION OF EDGE DETECTION

3.1.4 *Flood Filling*

The center of each region that was generated in the region growing stage is then marked as a seed point, and these seed pixels are used to carry out flood filling with the edges generated by Sobel edge detection used as boundaries. For each seed point, flood filling is carried out by first trying to fill a box centered at the point, with sides of an upper bound length. If this box stays within the region and does not hit any detected edge, then flood filling is applied recursively to the points surrounding the seed point. If the filled box does hit an edge, then boxes of decreasing length sides are used up to a lower threshold.

```
Flood_Fill(int x, int y, int n){
   // pix_t(x,y) : pixel at position x,y on image frame at time t
   // outline : array to indicate if pixel is an outline
   // moved : array to indicate pixel motion
   // box_n(x,y) : box of length n centered at x,y

   if n < threshold
     return

   for x,y ∈ set of seed points
     if outline[box_n(x,y)] == false
       moved[box_n(x,y)] == true
       Flood_Fill(x + n, y, n)
       Flood_Fill(x - n, y, n)
       Flood_Fill(x, y + n, n)
       Flood_Fill(x, y - n, n)
     else
       Flood_Fill(x, y, n-1)
     endif
   endfor
}
```

TABLE V. IMPLEMENTATION OF FLOOD FILLING

This procedure for each seed point, as given in Table V, serves to intersect information concerning broad regions which are thought to be moving, with information obtained from discerning complete edges of regions in the image, to produce whole regions which are moving in the image frame as can be seen in Fig 9.

3.1.5 *Bounding Box Filtering*

Once moving whole regions in the image have been obtained, a bounding box is determined which includes all generated regions. It is then checked if this box satisfies certain criteria to aid in determining if the motion detected is legitimate. First, the number of marked pixels in the box is checked and if they exceed some threshold value (in the case that flood filling has escaped a boundary of the object being tracked), the motion is disregarded. The ratio of the sides of the bounding box is then noted and if this ratio does not fall within certain limits we can also ignore the motion as we know that we will be tracking people who will have specific shape signatures. Lastly the bounding boxes in this frame and the last are compared and if the distance between their centers is greater than a given threshold, the motion is considered illegitimate as we are likely to be seeing some other object in this frame.

3.1.6 *Histogram Vector Dot Product Filtering*

At this point, we are fairly sure that the object we have detected is the desired one and we go on to generate an appearance model histogram model of the marked pixels in this frame. To produce this histogram, an array index value for each marked pixel in the bounding box is generated using its color value at reduced resolution by taking the highest n, p and q bits of the pixel's Y, U and V color values respectively and concatenating them, to produce an index of n+p+q bits. A frequency count of these values is then stored as an appearance model histogram as can be seen in Table VI.

```
Appearance_Histogram_Generation(){
   // pix_t(x,y) : pixel at position x,y on image frame at time t
   // moved : array to indicate pixel motion
   // histogram : array to act as frequency count of pixel types based on color

   for x,y where moved[pix_t(x,y)] == true
     index = concatenate(highest bits of color values of pix_t(x,y))
     histogram[index]++
   endfor
}
```

TABLE VI. GENERATION OF APPEARANCE MODEL HISTOGRAM

Once this appearance model histogram has been created, the dot product between the histogram produced from the current frame, and the base histogram produced from the frames up to this point (the generation of which we will see in the next section), is calculated. If the dot product between the two histograms is small (indicating dissimilarity), this would mean that although the moving object being observed resembles the desired target object (in terms of shape and size), the color profile of the object in this frame does not correspond with the object tracked in previous frames and so is discarded. This might be cause by lighting aberrations in the current frame or the fact that we have mistaken another similar object for our target.

3.1.7 Generate New Base Histogram

If the dot product between the histogram produced from the current frame and the base histogram produced from previous frames is higher than a threshold value (indicating similarity), then the object observed to be moving in this frame is considered to be the same one tracked in previous frames and it is used to update the base histogram characteristics. This is done by weighting the histogram by taking the pixel counts of the current frame into account as can be seen in Table VII.

```
Base_Histogram_Generation(){
   // pix_t(x,y) : pixel at position x,y on image frame at time t
   // moved : array to indicate pixel motion
   // base_histogram : cumulative histogram produced from previous frames
   // current_histogram : histogram produced from current frame

   for n ∈ base_histogram length
     base_histogram[n] = (α*current_histogram[n]) + ((1-α)*base_histogram[n])
   endfor
}
```

TABLE VII. GENERATING NEW BASE HISTOGRAM

When generating the new base histogram, the value of α is obtained by multiplying the dot product of the two histograms (obtained in Histogram Vector Dot Product Filtering) with the current acquisition potential. This ensures that in the process of adapting the base histogram to generate a final histogram model, frames with marked pixels which are similar to the existing base histogram influence it more heavily and thus downplays the effect of accidental motion (as we presume that the desired tracking object is predominantly moving in the scene). This process of base histogram generation also makes certain that we value frames at the beginning of the acquisition phase (when the acquisition potential is low and we are unsure of what we are tracking) to a lesser degree than frames towards the end of the acquisition phase (when acquisition potential is high and the base histogram model is somewhat accurate to our desired target).

3.1.8 Change in Acquisition Potential

Once the base histogram has been adapted to include the new frame, the acquisition potential changes similarly and increases by a factor proportional to the dot product of the two histograms (obtained in Histogram Vector Dot Product Filtering).

3.2 Target Tracking

Once the detected motion has increased the acquisition potential to the extent that it crosses some given threshold, the system transitions to the tracking phase with the evolved base histogram at that point being taken to be the final appearance model histogram, *m*, to be used in tracking. The bounding box enclosing all marked pixels in the last frame of the acquisition phase is also generated and the center of this box, *c*, indicates the position of the acquired target. In the next frame, a new histogram, *d*, is generated from all the pixels from the new frame which appear in the bounding box and the sample weight, *w*, of each pixel in this bounding box is calculated as given in (3).

$$w_i = \sqrt{\frac{m_i}{d_i}} \qquad i \in pixels \qquad (3)$$

A mean-shift vector is then generated as given in (4) and is used to translate the bounding box from the previous frame.

$$\Delta_v = \frac{\sum w_i |i - c|}{\sum w_i} \qquad i \in pixels \qquad (4)$$

The target is tracked throughout the frames until calculated sample weights increase past a certain threshold, at which point the target is considered to have been lost and the system reverts to the target acquisition stage.

4 TESTS AND RESULTS

The robustness and accuracy of the implemented person tracking system was evaluated by doing a visual inspection of how pixels were being marked, determining the sensitivity of the acquisition state via a confusion matrix, gauging the effectiveness of the acquisition potential, and identifying situations in which tracking was lost.

4.1 Progression of Pixel Marking During Acquisition Phase

A sequence of the marked pixels is given below to show the state of marked pixels as various stages of the target acquisition phase are applied to the image.

Figure 7. Original Frame (left) & Image Subtraction (right)

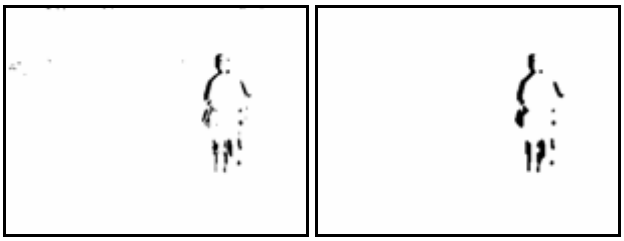

Figure 8. Application of Offset Filter (left) & Application of Morphological Filter (right)

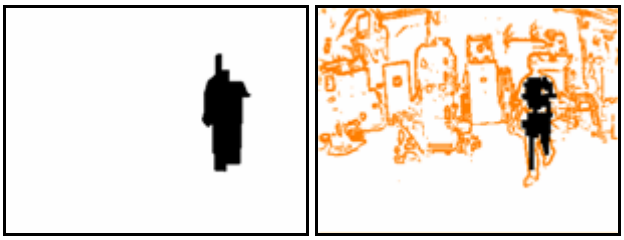

Figure 9. Application of Region Growing (left) & Intersection of Edge Detection and Flood Filling (right)

Visual inspection of the progression reveals that each step is effective in its role and the final result consists of marked pixels which are indeed indicative of the person being tracked.

4.2 Accuracy of Motion Detection During Target Acquisition Phase

Motion Detection accuracy was measured in two situations, namely with the robot balancing with no target to be tracked and in a normal tracking situation. The test was done by constructing a confusion matrix displaying the correctness of the motion detection algorithm in both situations.

For ground truth, a human operator recorded the presence or absence of motion that was meant to be detected for each frame of the sequence. Thus, the confusing matrix represents the detection results of the algorithm when compared to this hand labeled sequence. The following table shows the resulting confusion matrix in the two situations.

	% Frames in which Motion is Detected		**% Frames in which Motion is Undetected**	
	A	**B**	**A**	**B**
Motion Present in Image	0.0	21.9	0.0	52.1
Motion Absent in Image	0.5	7.4	99.5	18.6

TABLE VIII. CONFUSION MATRIX WITH NO TARGET (A), NORMAL TRACKING WITH SINGLE HUMAN TARGET (B)

From the low proportion of false positives, we can see that the person tracking algorithm can be adjusted to a level of sensitivity such that any vertical motion observed due to the rocking motion of the robot as it attempts to balance can be negated and only desired motion is considered.

4.3 Effectiveness of Acquisition Potential During Target Acquisition Phase

Utility of implementing the acquisition potential was tested by gauging the progression of the potential in three situations, namely with the robot balancing with no target to be tracked, when a person walks across the field of view of the robot without desiring to be tracked, and in a normal tracking situation.

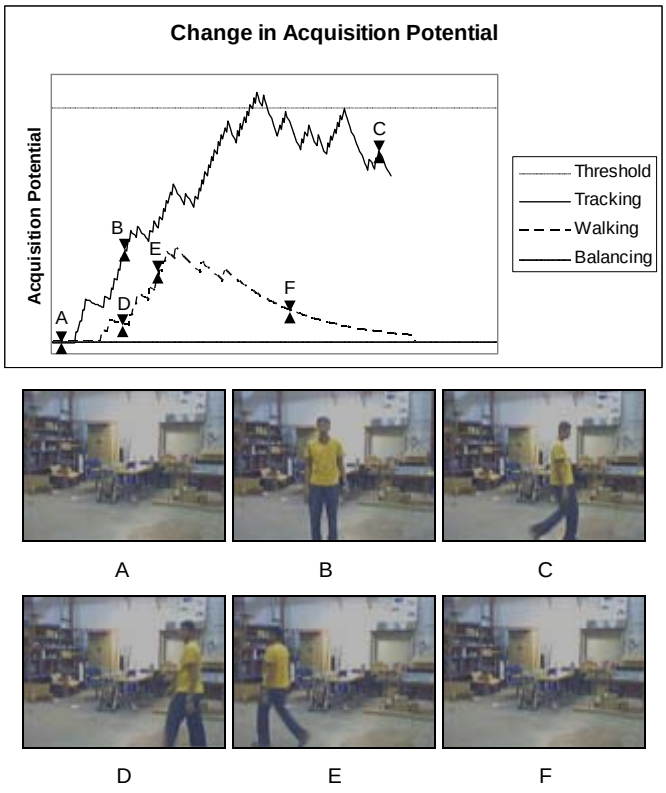

Figure 10. Change in Acquisition Potential

From the results in Fig. 10, we can see that implementation of the threshold potential does allow for the distinction between situations in which tracking is desired and those in which it is not. This distinction is crucial when trying to prevent potentially hazardous situations in which accidental target acquisition by the robot occurs.

4.4 Situations in which Tracking is Lost

Once the target has been acquired, the system will continue to track the target until sample weights when doing mean-shift calculations exceed a given threshold symbolizing that the target has changed. The situations in which this happens can be enumerated as follows.

- *Target Shrinks in Robot's Vision*

This occurs when the acquired target walks away from the robot fast enough that the robot cannot match the target's speed causing a considerable decrease in the number of pixels that the robot sees of the target. This leads to the robot no longer recognizing the target and reverting back to the target acquisition state.

- *Target Leaves Robot's Field of View*

This situation occurs when the target leaves the field of view of the robot due to instances such as when the robot is not able to turn at the sufficient speed to maintain sight of the target.

- *Change in the Color Characteristics of the Target*

This occurs when there is a fundamental change in the color values of the pixels which are in region thought to contain the target from the previous frame. This occurs when the target changes the color of the appearance he presents to the robot such as when he takes off his jacket to reveal a differently colored shirt, or when turning around while wearing a shirt whose front and back are of different colors.

5 Related Work

Temporal differencing (image subtraction) was used in [5], which involved calculating pixel-wise differences in intensity for 2 adjacent frames in the image sequence. Motion imaging was then generated using thresholding followed by clustering moving sections into motion regions and this method was implemented real time on a video stream obtained from a static video mount. For a balancing robot, where the camera is perturbed by small deviations as the robot balances, such temporal differencing would therefore generate false positives when determining motion in the sequence due to the rocking motion of the robot. An additional offset filter (mentioned in the Section 3) is applied to correct for this problem.

Three-frame differencing [1], is an approach used in video surveillance to overcome false positives being generated in the "holes" that appear in regions that an object moves away from in the image sequence. The three-frame differencing algorithm marks pixels as moving only if pixel intensity has changed significantly between both the current frame and last frame, and the current frame and next-to-last frame. We have not used a three-frame differencing approach as the algorithm does not account for additional false positives generated by image sequences from a camera on a moving, in this case balancing, platform. Instead we address this issue through the intersection of connected components based on homogeneity and motion.

6 Conclusion and Future Work

In this paper we have outlined some of the challenges involved when attempting to carry out person tracking on the Segway RMP platform. We have also presented a possible approach to implementing this task which involves a target acquisition phase consisting of various filters to obtain marked pixels indicating motion, and a target tracking phase in which the acquired target is tracked using the mean-shift algorithm. The accuracy and feasibility of the approach were also addressed with samples of the implemented solution given.

One possible improvement for the system involves extending the mean-shift algorithm to implement target tracking with changes in scale as mentioned in [1]. This extension of the algorithm would allow for increased tracking accuracy even when the target walks away from the robot quickly enough to cause a considerable decrease in the number of pixels the robot sees of the target.

6.1.1.1.1 Acknowledgments

This research was sponsored by the United States Army under Grant No. DABT63-99-1-0013. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of DARPA, the US Army, or the US Government.

6.1.1.1.2 References

[1] *R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, "A System for Video Surveillance and Monitoring: VSAM Final Report," Technical report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, May, 2000.*

[2] *R. Collins. "Mean-shift Blob Tracking through Scale Space", Computer Vision and Pattern Recognition (CVPR'03), IEEE, June, 2003*

[3] *R. Gonzalez and R. "Woods Digital Image Processing", Addison Wesley, 1992, pp 414 - 428.*

[4] *D. Vernon. "Machine Vision", Prentice-Hall, 1991, Chap. 5.*

[5] *A. Lipton, H. Fujiyoshi and R. Patil. "Moving Target Classification and Tracking from Real-time Video" IEEE Workshop on Applications of Computer Vision (WACV), Princeton NJ, October 1998, pp.8-14.*

[6] *M. Montemerlo, W. Whittaker, and S. Thrun, S., "Conditional Particle Filters for Simultaneous Mobile Robot Localization and People-Tracking", IEEE International Conference on Robotics and Automation (ICRA'02), Washington, DC, 2002.*

[7] *H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. "RoboCup: A Challenge Problem for AI and Robotics", RoboCup-97: Robot Soccer World Cup I, Nagoya, L.N. on A.I.,Springer Verlag, 1998, 1-19.*

[8] *B. Browning, P. Rybski, J. Searock, and M. Veloso. "Development of a soccer-playing dynamically balancing mobile robot",IEEE International Conference on Robotics and Automation (ICRA'04), in press.*

[9] *J. Bruce, M. Bowling, B. Browning, and M. Veloso. "Multi-Robot Team Response to a Multi-Robot Opponent Team". IEEE International Conference on Robotics and Automation, Taiwan, May 2003.*

[10] *M. Nicolescu, M. J Mataric, "A hierarchical architecture for behavior-based robots", First International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, ITALY, July 15-19, 2002.*

[11] *J. Searock, B. Browning, and M. Veloso. Turning Segways into Soccer Robots. In In Proceedings of IROS'04, September 2004.*

[12] *I.Haritaoglu, D.Harwood, and L.Davis. A Real Time System for Detecting and Tracking People, In FGR98, 1998.*