Detection and Tracking of Human Subjects

a thesis
in STS 402

presented to

the faculty of the
School of Engineering and Applied Science
University of Virginia

in partial fulfillment
of the requirements for the degree

Bachelor of Science in Computer Engineering

By

Douglas Grosvenor

April 30, 2009

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for papers in STS courses.

signed _____
Douglas Grosvenor

approved _____ date _____
Kevin Skadron, Department of Computer Science

approved _____ date _____
Peter Norton, Department of Science, Technology, and Society

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

ABSTRACT

The goal of the thesis project was to devise an algorithm to detect and track people in a static video. Existing techniques are inadequate; instead a new approach based on background subtraction is used. The approach is successful with a static camera. In background subtraction, the background of the video is calculated a priori and then subtracted from each frame of the video. This isolates the foreign objects, which are detected via two simple algorithms. Both algorithms are based on the subject's center of mass, but the first algorithm traces the path of the person around the video, making it very cluttered. This implementation was not very resilient to noise so a new algorithm was developed which tracked the person by placing a green rectangle around them. In the end, this project successfully detects and tracks a person walking for the duration of the video while displaying it to the user. Since this is only a proof of concept and the detection and tracking did not run in real-time, further work is needed to improve the algorithm. One suggested method to enhance this algorithm involves porting the code over to CUDA to take advantage of the inherent parallelism in image manipulation.

REPORT INTRODUCTION

Background

A. Problem Statement

Image segmentation is the process by which an image is broken into similar pieces or in which desired parts of an image are extracted and processed. Over the past fifteen years or so, researchers have become interested in automating this process to expedite slow and tedious manual processes, which in some cases require "tens of hours of user interactive image processing" (Ray and Acton, 2004). As this automation progressed, various disciplines, ranging from medicine to robotic vision, became interested in the benefits image segmentation offered. However, many of the algorithms generated are application specific. I had a first-hand experience of this when I tried using two image-segmentation algorithms. Since no preexisting technique to detect and track a person in a video image met my standards, I had to develop my own.

B. Image Segmentation

Even though some users develop their own image segmentation algorithms, they all still derive many of the same benefits from their use; including decreased processing time. Doctors on one hand appreciate the use of image segmentation because of a brain segmentation technique called "stripping." According to Rehm, "`Stripping` is segmentation that classifies head image elements into two rigid classes: brain and non-brain" (2002). By "stripping" an MRI, doctors can diagnose, treat, and visualize their patients' problems accurately and quickly (Rehm, 2002). Another area in which doctors use image segmentation is image-guided surgery. According to O'Donnell:

> In order to remove brain tumors or to perform difficult biopsies, surgeons must follow complex trajectories to avoid anatomical hazards such as blood vessels or functional brain areas. Before surgery, path planning and visualization is done using preoperative MR and/or CT scans along with three-dimensional surface models of the patient's anatomy (2001).

With this segmentation, surgeons better understand where they need to go and what they need to do.

Image segmentation is also useful in satellite imagery, computer vision and security. Researchers can use image segmentation in "remote sensing applications ranging from environmental and agricultural to national security interests" (Rekik et al, 2009). With computer vision, robots and other devices can discern objects and navigate around them (Kubik and Sugisaka, 2003). Many applications in security involve image segmentation, including facial and fingerprint recognition, and detection and tracking of people by security cameras. The latter is the main focus of this paper. Real world applications include detecting and tracking people in airports and monitoring events at home. Eventually, when these technologies reach maturity, we will use them with security cameras to track and arrest criminals.

C. Detection and Active Contour

Many image segmentation algorithms have been proposed. One of the better-known algorithms is an edge detection based gradient vector field (GVF) active contour. Since each image is composed of pixels, and each pixel is represented by a number associated with the color it represents, an algorithm can be created that determines the

gradient (the difference between neighboring pixels). This type of algorithm generates an edge map (Figure 1), along with its respective grayscale image. Once researchers manipulate this edge map into a more useable form, they then apply an active contour, also known as a snake, to convolve around the image, trying to find any defining shape. This type of snake was used in the leukocyte detection and tracking system of Boyer et al., a system upon which this thesis was based (2009). Boyer et al. also implemented their code in NVIDIA's Compute Unified Device Architecture (CUDA).

D. The GPU and CUDA

CUDA is a relatively new programming language which takes advantage of the GPU as a math coprocessor. Since our insatiable appetite for more processing power has driven us near the edge of Dennard Scaling (Dennard et al., 1974), we are getting to a point where we cannot keep increasing the clock speed of our processors. To alleviate this problem, we started adding more cores. However, we can already take advantage of the numerous devices out there that have a high degree of parallelism. Many of these devices are application specific, like DSPs; however, one of these massively parallel devices was created for other purposes but has relatively recently been used for general purpose processing. This device, the GPU, has been enhanced and developed from the gaming industry. Because of the demands it has placed on graphics cards, their processing capabilities have doubled every 6 months (Fan et al, 2004). Even though CUDA and general-purpose computing on the GPU (GPGPU) can significantly improve performance, performance gains can only be seen in highly parallelizable applications.

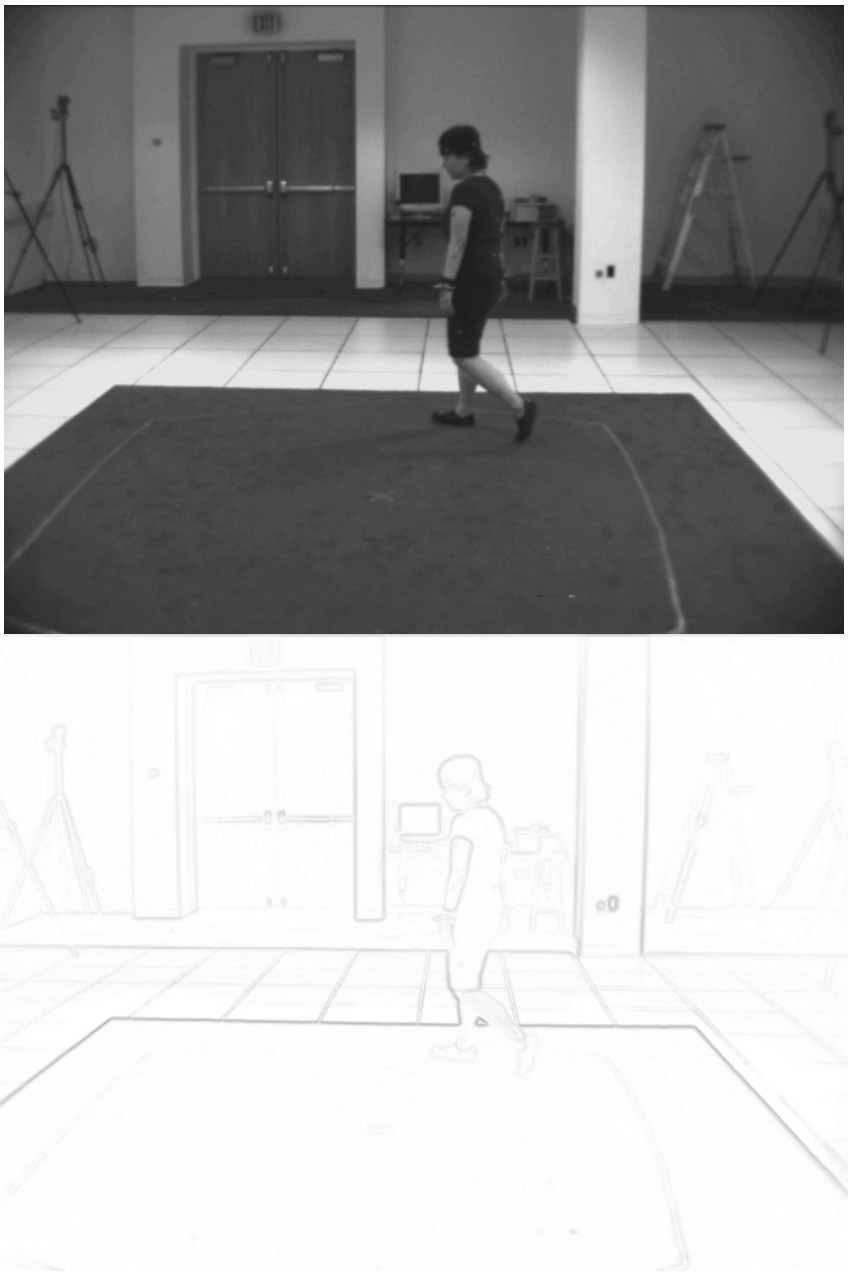

Figure 1. Frame and its Edge Map
(Top) A grayscale image. (Bottom) The gradient, or edge map, of the same frame. Notice the defining edges of all of the objects in this picture.

GPGPU cannot yet completely replace CPUs, but will be used in conjunction with them, creating a heterogeneous computing infrastructure.

Related Work

A. Leukocyte Detection and Tracking

This thesis project began as a development upon earlier work to detect and track leukocytes in the blood stream. Since an immune response can be deduced from the interaction of the white blood cells and the cell walls, doctors want to track these cells to improve diagnosis and treatment regimens (Boyer et al, 2009). This detection and tracking system creates a gradient inverse coefficient of variation (GICOV) to determine the edges around the leukocyte cells. After detecting these edges, a motion GVF is created to convolve a snake to outline and detect the leukocytes. However, as I would later find out, this algorithm, while it could work for the problem I am trying to solve, is needlessly complex for this problem.

B. Vector Field Convolution

Along with the leukocyte detection and GVF tracking system, a new snake algorithm was developed that is supposedly faster than current active contours, known as vector field convolution (VFC). The GVF snake "required 3 to 10 times more computational expense than the VFC" (Li et al, 2007). Given an assortment of images, ranging from very noisy images to peculiar starting locations for the snakes, the VFC was able to conform to many more images then the GVF (Li et al, 2007).

Organization of Paper

This paper explains how to create an algorithm that can detect and track a moving person through a still camera. I will also review the many obstacles that I faced to create

this algorithm. This algorithm uses a walking video as the test file and a generated or selected background image to detect any alien objects in the current frame. Even though this is computationally simple, since it is across numerous elements the whole algorithm cannot run in real time. However, I provide recommendations that might speed up the algorithm enough for real-time use and analysis.

METHODS

Brick Walls

"If at first you don't succeed, try and try again" would be an apt philosophy to apply to this thesis project. Initially this project was designed to take a preexisting code base that detected, and potentially tracked, objects in a systems biology space, and speed the code up with CUDA. However, from the onset I ran into many problems. Originally I was going to use the VASARI Image Processing System (VIPS) for the image segmentation and detection, but after analyzing the runtime characteristics of the algorithm, I found that VIPS was a poor candidate for porting to CUDA. I next tried to use a preexisting code base that detected and tracked leukocytes (Boyer et al., 2009). However, instead of detecting and tracking objects in a systems biology application, this algorithm would now detect and track human subjects. Boyer's code was originally derived from Matlab, which he ported over to C and CUDA to speed up the algorithm. After I modified his code to work with the provided test video, however, the algorithm did not prove to work as expected. I then consulted my technical advisor, Kevin Skadron, and discussed the requirements Scott Acton wanted in the algorithm. I decided to try a different type of detection system. In his code, Boyer used a GICOV detection system

and a GVF active contour; however, the new detection system used a vector field convolution (VFC) active contour. And, like all of the previous solutions, it did not work out as expected. Finally I received some help from a number of electrical engineering graduate students. Alla Aksel, Saurav Basu, and Josh Poulin, helped me immensely. After talking with them I had a better understanding of what I needed to accomplish and how I should do it.

Detection: The Light at the End of the Tunnel

With setback after setback, I was finally making significant progress. Originally, I wrote most of the code directly in C to make it easier to port over to CUDA, but I eventually realized that programming everything in Matlab would be easier initially.

With all of my algorithms I had difficulty deciding how to determine that a person was in the image. Since a computer represents everything in binary, it cannot distinguish objects in the image. To help the computer detect foreign objects in an image, I had to manipulate the image in such a way to remove everything in it except for the object(s) I want to track. To make it easier for the computer to detect any foreign objects in the image I first had to have a reference image that represented everything that would remain static in the image. This reference image was called the background image, and the original background image for the algorithm was actually skewed slightly, throwing off my results.

I eventually fixed this problem by creating my own background image from the video file provided. I did this by iterating over all of the frames in the video and storing every fifth frame's pixel value. I selected every fifth frame because the frame rate was

sixty frames per second (fps) and the video was twenty-one seconds long, requiring substantial amount of memory to store all of the pixels, too much for Matlab. However, this did not hinder me from creating the background image. Once I had arrays of stored pixels values from different frames, I took the median pixel value from across all of the frames to retrieve the most normal pixel value. The median was more appropriate than the mean because the mean would take into account any objects that had disrupted that pixel. Instead, by taking the median pixel, I get a pixel value that was seen in the frame, and since the person is walking around, it was almost guaranteed that there were more background pixels in each array then pixels of the person. Once I had this array of median values, I saved the image that it created, then I had a perfect background image (Figure 2).

Figure 2: Background Image Generated from the test video

With this background image I now started the main process of isolating the person from the rest of the image. Since the video contained frames of a person walking around the room with the same background as the file I created, isolating them from the rest of the image was straightforward. To make the image a little easier to handle and work with on the computer, I first converted every frame from the RGB color scheme to grayscale. This reduced the amount of information I worked with, making the whole algorithm faster. The first frame of the video can be seen in Figure 3, first in the RGB color and then as a grayscale image. Once I had this representation of the image I needed to get rid of everything except for the woman, and the easiest way to do this is to just subtract the generated background image from the current frame. However, we do not want just to subtract the two images, we need to take the absolute value to take into account the negative values that might be generated; the generated image can be seen in Figure 4. Notice in this image that some locations on the subject's body are darker than others. This effect is due to the fact that she is wearing clothes that are similar to the background color and they almost create a pixel perfect match, subtracting to almost zero; this effect will show up more prominently in later images. Once we have this image, the computer can determine that there is an alien object in the reference frame. To make her location in the image more pronounced, however, we need to create a binary image. To do this I had to determine an appropriate threshold at which to cut off the grayscale values. Initially I used Matlab's 'graythresh(i)' function to generate the threshold for the image. However, I later manually tried different values and realized that I was getting better results with a predetermined threshold than a dynamic one. This was because the only differences between the images were any foreign objects, the woman, or any lighting differences,

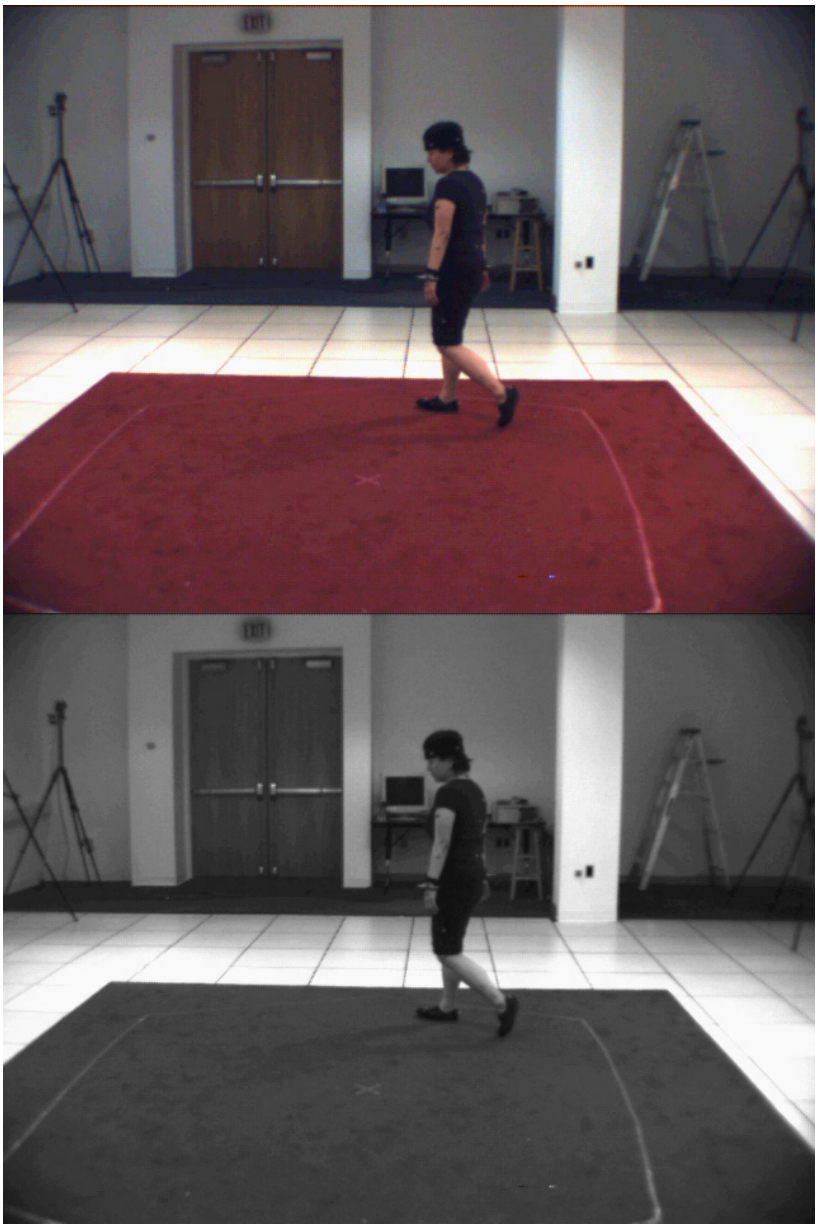

Figure 3: RGB and Grayscale Frame
(Top) First frame of the walking video in original color. (Bottom) Same frame converted to grayscale.

which would be very minimal. With 'graythresh(i)' the threshold would change from image to image but it would stay around 25% to 35%. However, I experimentally determined that a threshold around 8% would generate a better representation of the woman while getting rid of most of the noise in the image. I then used 'im2bw(image,level)' to generate the binary image (Figure 5). Now that I had this perfect

Figure 4: Subtracting Background Frame from Current Frame
The same frame after subtracting the background frame from the current frame. Notice the subtle lighting differences at her feet where her shadow is cast.

representation of where the woman was in the image, I could have the computer track her wherever she moved.

Tracking: Follow my finger…. Or that big green rectangle around the person…

With detection accomplished, all that was left was tracking and displaying the subject's location. I considered doing a center of mass on her and have that track her around the screen. However, after a while you could not tell exactly where she recently was because there were many tracked points on the screen. Also, by doing only a center of mass tracking and displaying system, the system was less resistant to noise. This is because the center of mass was generated by taking the

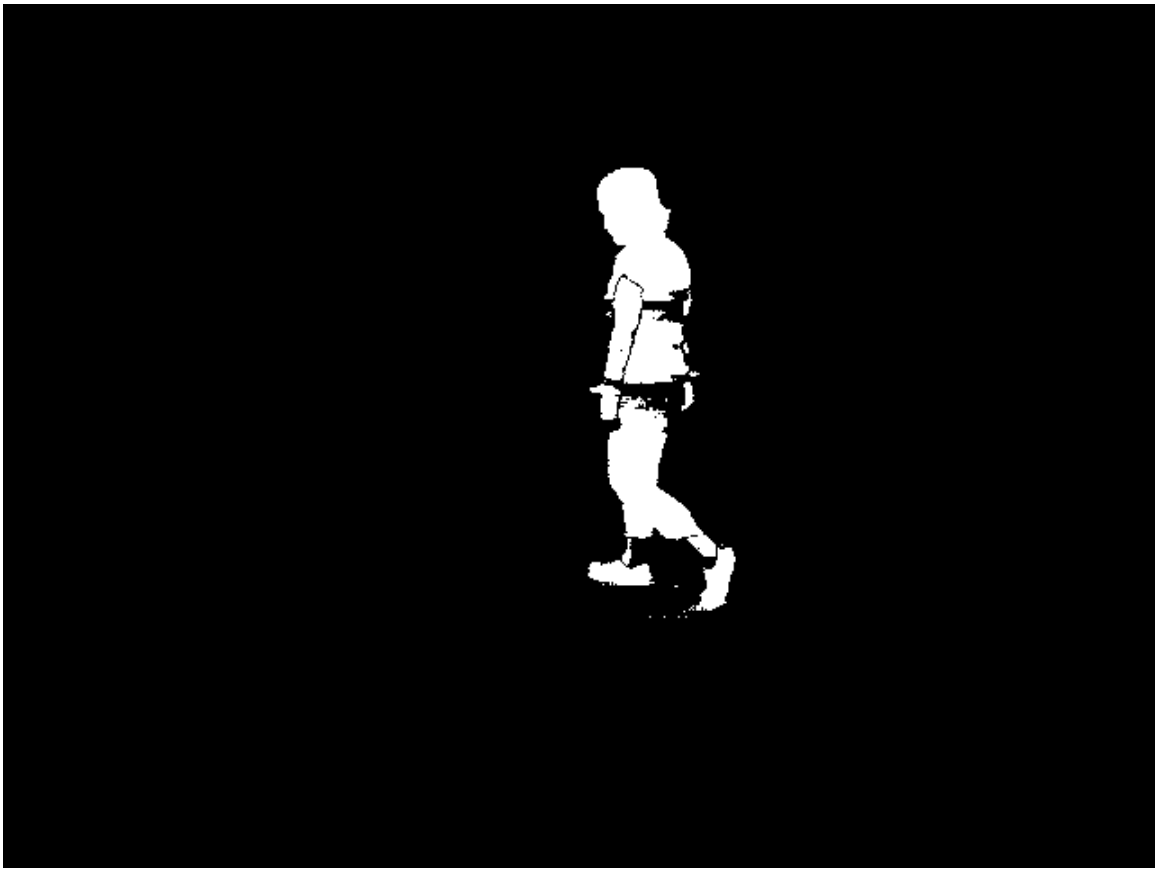

Figure 5: Binary Image
Binary image generated with a threshold of 8%. Here you can see where some of her body blends in with the background image.

average of the x and y coordinates of the person's location, represented by the white pixels. If there was any noise around the image, a pixel switching from black to white, it could throw off the center of mass significantly. So instead, Basu came up with an interesting alternative that was more resilient to noise and defined the person better. This algorithm would create an x-y axis with the center at the center-of-mass, splitting the person into four quadrants. Then the algorithm would average the x and y coordinates again in each quadrant, creating a point in each quadrant. These points were then connected together to create a rectangle that would follow the person around. The center-of-mass implementation can be seen in Figure 6 and the rectangle method can be seen in Figure 7.

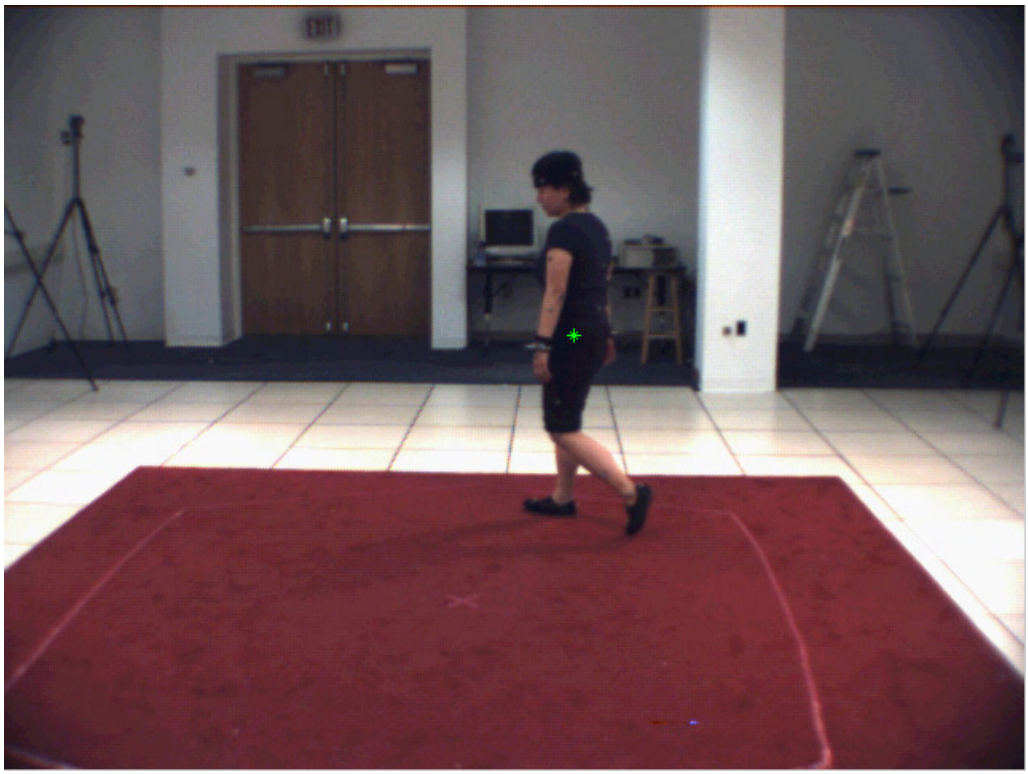

Figure 6: Center-of-Mass
The first frame with the center-of-mass displayed.

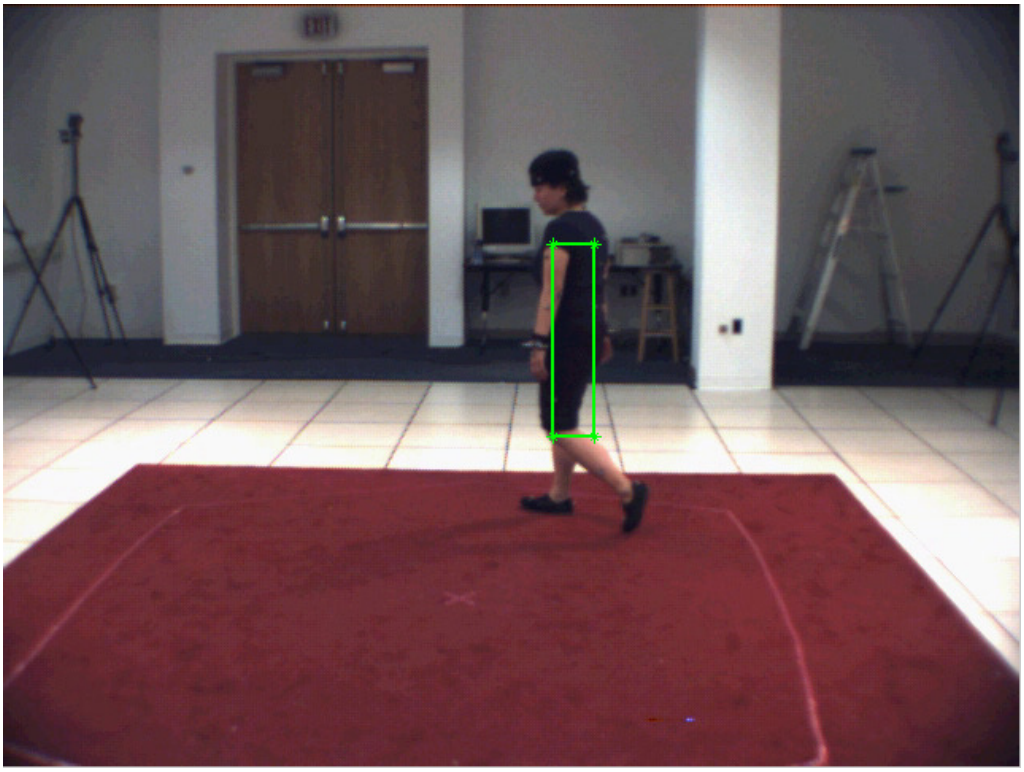

Figure 7: Tracking Rectangle
The same frame with the rectangle visualization

CONCLUSION

As I have demonstrated, this is a relatively easy and fast way of detecting and tracking a person in a video. Given that the video is twenty-one seconds long at sixty fps, the algorithm took a total of 123 seconds to process complete. While this is not real time, it can process approximately ten fps, which is relatively close to the normal video frame rates of thirty fps. However, since this is a proof of concept, significant speedup can be achieved later. If I kept using Matlab to analyze videos and track people, I would skip the detection and tracking phase for every frame. Instead, I would detect and track every other frame or so to decrease the run time while still accurately representing what is happening. However, one particular solution is available to get a more accurate detection and tracking system. Since this algorithm is mainly matrix manipulation, it is an excellent candidate for CUDA, which was my original intention until I ran into so many problems. My only reservation porting over to CUDA is that the number of transfers to and from the GPU may cause some problems. Yet because these images are less than one MB, it is unlikely the application will be limited. However, even with these optimizations, the algorithm needs.

As we have seen, this algorithm will detect and track a person in a video; however, this is only one person. If there was another person or alien object, the detection and tracking would be skewed significantly. To remedy this, the algorithm would need to isolate each independent object in the binary image and track it there. Even with this implementation there is a possibility of two overlapping objects combining into one. Preventing this would require memory as to where each object was before, in which direction it was moving, and which one was in front of the other.

BIBLIOGRAPHY

Boyer, M., Tarjan, D., Acton, S. T., and Skadron, K. (2009). "Accelerating Leukocyte Tracking using CUDA: A Case Study in Leveraging Manycore Coprocessors." In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, May 2009, to appear. Retrieved March 17, 2009 from http://www.cs.virginia.edu/~mwb7w/publications/ipdps09.pdf

Dennard, Robert H., Gaensslen, Fritz H., Yu, Hwa-Nien, Rideout, V. Leo, Bassous, Ernest, and Leblanc, Andre R. (1974). IEEE Journal of Solid-State Circuits, Volume: 9, Issue: 5. Retrieved April 21, 2009 from http://www.eng.auburn.edu/~vagrawal/COURSE/E6270_Spr09/READ/Design%20of%20ion-implanted%20MOSFETs%20with%20very%20small%20physical%20dimensions.pdf

Fan, Zhe, Qiu, Feng, Kaufman, Arie, and Yoakum-Stover, Suzanne (2004). GPU Cluster for High Performance Computing. ACM / IEEE Supercomputing Conference 2004. Retrieved March 17, 2009 from http://www.cs.sunysb.edu/~vislab/papers/GPUcluster_SC2004.pdf

Kubik, Tomasz and Sugisaka, Masanori (2002). Image segmentation techniques and their use in artificial life robot implementation. Artificial Life Robotics. Retrieved March 17, 2009 from http://www.springerlink.com/content/h0p2k66k62787535/fulltext.pdf

Li, Bing and Acton, Scott T. (2007). Active Contour External Force Using Vector Field Convolution for Image Segmentation. IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 16, NO. 8. Retrieved April 17, 2009 from http://viva.ee.virginia.edu/publications/VFC_jour.pdf

O'Donnell, Lauren (2001). Semi-Automatic Medical Image Segmentation. Retrieved April 17, 2009 from http://people.csail.mit.edu/lauren/publications/odonnell_ms_thesis.pdf

Ray, N., and Acton, S. T. (2004). Motion gradient vector flow: An external force for tracking rolling leukocytes with shape and size constrained active contours. *IEEE Transactions on Medical Imaging*, vol. 23, no. 12. Retrieved March 17, 2009 from http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01362749

Rekik, Ahmed, Zribi, Mourad, Hamida, Ahmed Ben, and Benjelloun, Mohamed (2009). Retrieved March 17, 2009 from http://www.waset.org/ijsp/v5/v5-1-5.pdf

Rehm, Kelly (2002). Medical Image Segmentation. Retrieved April 17, 2009 from http://www.neurovia.umn.edu/home/kelly/Course_notes/BPHY8148/2002/image_segmentation_lec2_forweb.pdf