COFFEE QUEUE PROJECT

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

George Dimitrov Gargov

March 2016

COMMITTEE MEMBERSHIP

TITLE:	Coffee Queue Project
AUTHOR:	George Dimitrov Gargov
DATE SUBMITTED:	March 2016
COMMITTEE CHAIR:	John Oliver, Ph.D. Associate Professor & Director of Computer Engineering Program
COMMITTEE MEMBER:	Jane Zhang, Ph.D. Professor & Graduate Coordinator of Electrical Engineering
COMMITTEE MEMBER:	John Saghri, Ph.D. Professor of Electrical Engineering
COMMITTEE MEMBER:	Wayne Pilkington, Ph.D. Associate Professor of Electrical Engineering

ABSTRACT

Coffee Queue Project

George Dimitrov Gargov

In this paper, a computer vision system for counting people standing in line is presented. In this application, common techniques such as Adaptive Background Subtraction (ABS), blob tracking with Kalman filter, and occlusion resistive techniques are used to detect and track people. Additionally, a novel method using Dual Adaptive Background Subtractors (DABS) is implemented for dynamically determining the line region in a real-world crowded scene, and also as an alternative target acquisition to regular ABS. The DABS technique acts as a temporal bandpass filter for motion, helping identify people standing in line while in the presence of other moving people. This is achieved by using two ABS with different temporal adaptiveness. Unlike other computer vision papers which perform tests in highly controlled environments, the DABS technique is tested in a crowded Starbucks© at the Cal Poly student union. For any length of people standing in line, result shows that DABS has a lower mean error by one or more people when compared to ABS. Even in challenging crowded scenes where the line can reach 19 people in length, DABS achieves a Normalized RMS Error of 43%.

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

Deciding the optimal time to get a cup of coffee at conveniently located coffee shops can be a real hassle. Go at the wrong time, and you could be faced with a long line to wait in. This paper proposes a solution to alleviate this. Simply by checking a website, coffee aficionados can see the line size, all without violating privacy of the people waiting in line.

The system that makes this solution possible is made up of several parts, as shown in **Figure 1.1**. First, a webcam is placed near the coffee shop, facing the line for coffee. Next, the video feed is transmitted to a server where the computer vision system, discussed in this paper, counts the number of people in line. The system outputs the number of people in line, and a confidence value to a database. This data is then displayed on a website for any customer to see.

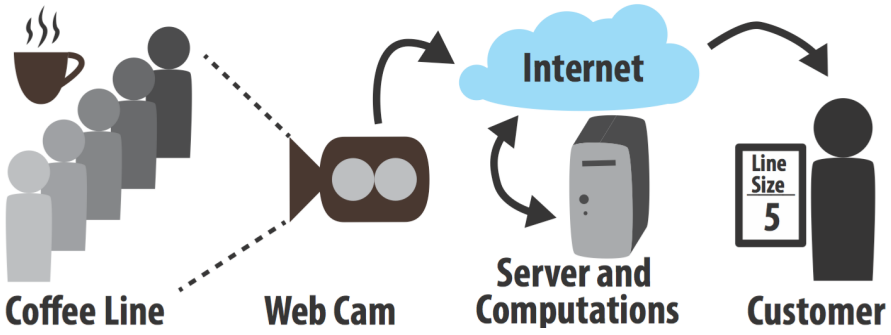


Figure 1.1: System overview

In its most abstract, the computer vision system presented here can be broken down into three stages, as shown in **Figure 1.2**. The first stage is target acquisition, where people are identified in the scene. The second stage is target tracking, where people's location and velocity are monitored and predicted. The third stage is the application specific stage, where the location of every person is used to determine who is in line. There are many computer vision techniques available for each stage of the

system. In order to confine the search space of viable techniques, several design requirements must be put in place.

Figure 1.2: Three stages of the system

The system should be as autonomous as possible, in order to minimize manpower during deployment to various locations. This means minimal user input or training data. While computer vision systems that leverage machine learning can produce powerful results, they are too reliant on good training data; therefore, it can be difficult to determine if a system underperforms from poor design or if it has been inadequately trained. In addition, the system must be capable of working in virtually any environment. If a web camera can be placed in this environment, the system must be able to process the video feed, and extract meaningful results. Lastly, the system must be able to count people in real-time. Regardless of the frame rate used for computations, a result should be available every couple of seconds.

2. Computer Vision System

From the aforementioned design requirements and a thorough examination of competing techniques, the system presented in this paper can be examined, one stage at a time. The general procedures performed in each of the three stages are shown in **Figure 2.1** below. OpenCV and C++ are used to implement these three stages in a pipeline multi-thread design, but this will be discussed later in section **3. Implementation**.

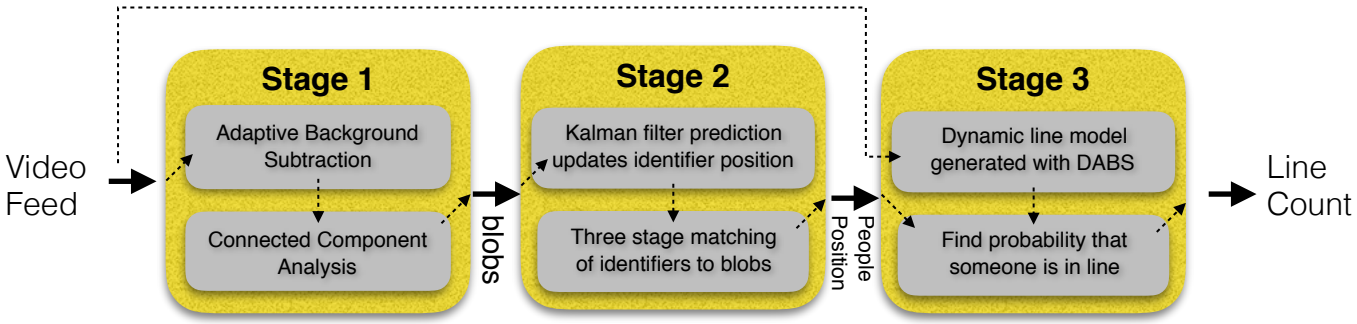


Figure 2.1: Three stages of the system in greater detail

In Stage 1 of the system shown in **Figure 2.1**, Adaptive Background Subtraction (ABS) [3][4] is used to find foreground pixels. Connected Component Analysis groups adjacent foreground pixels together to create blobs. This is the basis of target acquisition. An alternative technique is also proposed by this paper for targeting stationary people using Dual Adaptive Background Subtraction (DABS), and even more options such as HoG detection[5] are discussed in section **6.1 Target Acquisition Related Works**. Target acquisition will be covered in greater detail in section **2.1 System Stage 1: Target Acquisition**.

Then, in Stage 2 of the system depicted in **Figure 2.1**, a layer of identifiers is created for blob tracking. Their purpose is to be matched up with blobs from frame to

frame. Kalman filters are used to predict where blobs will be in the next frame. A three stage matching process[1] is used to find the best match for identifiers and blobs. First is one-to-one blob to identifier matching, next is many-to-one in the event of blob fragmentation, and, finally a one-to-many blob to identifier matching in the event of occlusion. Improvements are made to the stability of identifiers. This will be covered in greater detail in **2.2 System Stage 2: Target Tracking**.

Finally, in Stage 3 of the system presented in **Figure 2.1**, a dynamic line model is created using the novel DABS technique. The model is an image where every pixel is a probability that a line is formed in that location. The dynamic line model is then used in conjunction with the location of every person to determine the probability that someone is in line. These probabilities are then used to compute the final line count, which is outputted by the system. All this will be covered in greater detail in section **2.3 System Stage 3: Dynamic Line Model.**

2.1 System Stage 1: Target Acquisition

To find regions of interest that can be classified as targets, adaptive background subtraction (ABS) is used. Background subtraction, or foreground extraction, is done on a per pixel basis using a mixture of gaussians over a user specified temporal window. This allows for an ever-evolving model of the background that adapts to the scene changes during the day. The foreground is obtained by subtracting the background model from the current frame and applying a threshold. This produces a binary image with white pixels representing the foreground and black representing the background. A video sequence of the background, raw video, and foreground pixels can be seen in **Figure 2.2** below. Notice how when the people appear in the upper left corner of the video in the middle row of **Figure 2.2**, the adaptive background in the top row remains

unchanged, but the people are detected as part of the foreground in the bottom row of **Figure 2.2**.

Typically large impulse differences from the current frame and the background model are labeled as foreground. A person walking across the scene, for example, would trigger such an impulse and would therefore be classified as foreground. More sophisticated techniques such as those presented by Zoran Zivkovic [3][4] allow for gentle background variations from falsely being classified as foreground. For example, the sway of trees in the wind would not be picked up because the background variation is not significant enough to be classified as foreground. In addition, shadows can also be detected, albeit, this is largely dependent on the environment.

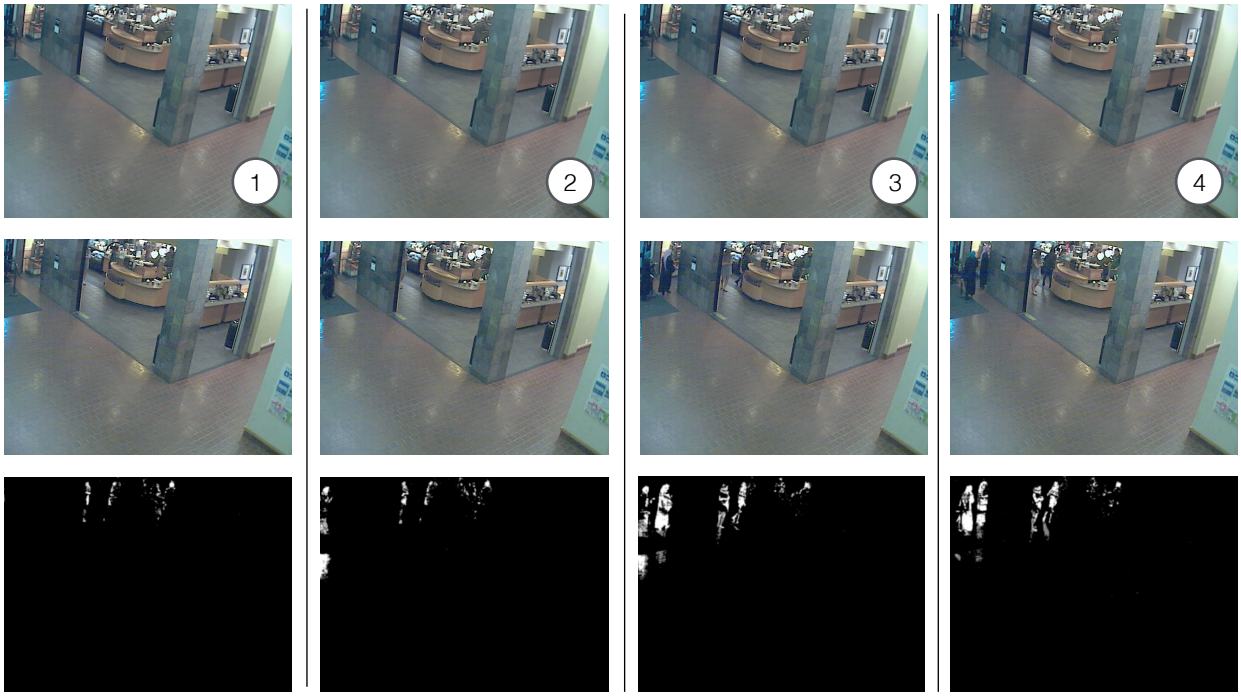


Figure 2.2: Adaptive background subtraction sequence. Top row is the adaptive background model, middle row is raw video frame, bottom row is the foreground or the difference between the raw video frame and adaptive background model

The most important parameter of ABS is its adaptiveness, meaning how slow or fast objects standing still fade into the background. Having a slow adaptiveness is beneficial for allowing people to stand still for a little bit before blending into the background. However, this makes the background model less resilient to noise such as false positives caused by changes in lighting, or movement of furniture, which would falsely be picked up as foreground and take longer to be absorbed back into the background model. Although some noise in the foreground can be observed primarily in the form of imperfect foreground pixels, this topic will be covered more in depth in section **4. Challenges and Limitations of the System.** Conversely, having a faster adaptiveness makes the background more resilient to noise, but makes pedestrians fade into the background faster.

Having a binary foreground image, the next step is to group adjacent foreground pixels together to create a set of blobs representing targets; this is achieved by a two-pass algorithm called connected component analysis. Minor filtering is performed at this stage. Only blobs that are comprised of over a certain number of pixels are permitted to the set of blobs representing targets. This filters out really small blobs that are essentially noise. At this point, the problem of occlusion becomes very apparent. In the ideal scenario there would be a strict one to one relationship between people and blobs. However when two or more people are close together or occluding one another, this shows up as one large blob. Therefore, higher level analysis is needed in order to overcome the problem of blob occlusion as well as identifying the location of every pedestrian in a scene to determine who is in line or not.

2.2 System Stage 2: Target Tracking

The identifier[1] provides a more reliable metric of location, in addition to making blob velocity measurable. In the most basic terms, identifiers are objects that are matched up with blobs on a frame-by-frame basis and serve as a container for blob metrics. An identifier has a location (x, y coordinates), velocity, dimensions (height and width), rate of dimension change and an “age.”

Identifiers are “born” when a blob in the set of indexed blobs does not have a good match to an existing identifier. This new identifier has the location and dimensions of the unmatched blob that caused it to be born. Its velocity, rate of dimension change, and age are set to zero. An Identifier “dies” if it remains unmatched with a blob for a certain amount of consecutive frames. Unmatched identifiers update their position based on their current velocity.

The Kalman filter is a critical part of the tracking layer because it provides a state to blobs that are inherently stateless. In this paper, a first order (position, velocity) model is used for the Kalman filter. Each identifier stores priori position,velocity, and dimension information necessary for the Kalman filter to make predictions. These location and dimension predictions made by the Kalman filter are used to establish nearest neighbor matches between identifiers and blobs. After all matches are made between blobs and identifiers, position and dimension information is used from the matched blob as measurement information to update each identifiers Kalman filter matrices. **Figure 2.3** below shows an overview of how the Kalman filter is used during the matching process of identifiers to blobs.

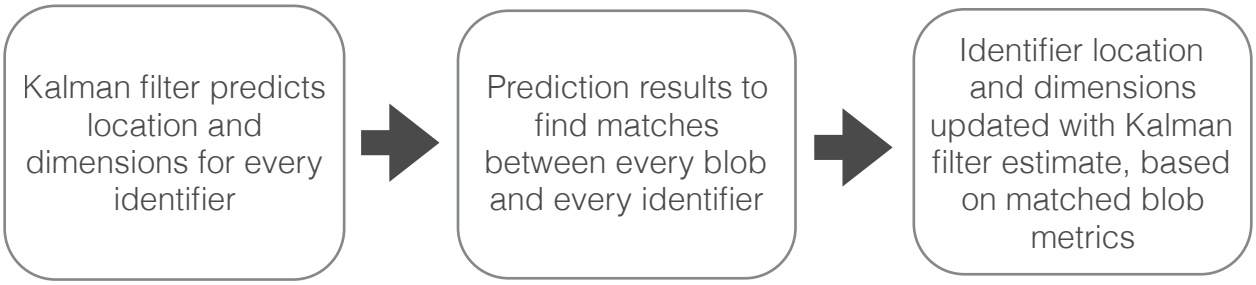


Figure 2.3: Overview of the matching process

OpenCV provides an implementation of the standard Kalman filter. To use it, basic knowledge is necessary and the steps to make the filter work are as follows: first, it is specified that 4 dynamic parameters are used, position (x,y point) and velocity (x,y vector). Next, it is specified that there are 2 measurement parameters, position (x,y point). Lastly, to keep things simple, no control data is used. Before the Kalman filter can be used to make predictions and take in measurements, several matrices must be specified. The measurement matrix is defined so only position data is kept ([1 0 0 0 ; 0 1 0 0]). The process noise covariance matrix, the measurement noise covariance matrix, and the error covariance matrix are set to an identity matrix with a tuned value between zero and one.

To tune the Kalman filter, several test videos of pedestrians walking were used. The three covariance matrices have their values adjusted so the matching process is maximized. It is difficult to tune two Kalman filters at the same time when the performance of either can affect the higher level goal of maximizing matches. Also, the performance of the filter would vary from video to video and even from blob to blob. This is because there is a certain amount of randomness that cannot be eliminated. At some point in the tuning process, the Kalman filters performance was deemed “good enough” for each of the test videos. Any detailed tunings of the Kalman filter would require modeling the motion of the scene to identify common cases, which is out of scope.

2.2.1 Blob Metrics

It is important to establish blob metrics before we can talk about scoring equations for calculating the best blob match for an identifier. First, a blob has a centroid (or center of mass), a height and width (dimensions), and central points [1]. Central points are local maxima of the distance transformation for a given blob. The distance transformation of a blob is the shortest distance from any “inside” pixel to any “outside” pixel. For example, imagine a circular blob of radius 5 pixels. The distance transformation of this circle would look something like in **Figure 2.4,** where the outer most pixel ring will be of distance 1 to to the “outside” and the inner most pixel ring will be of distance 5 to the outside.

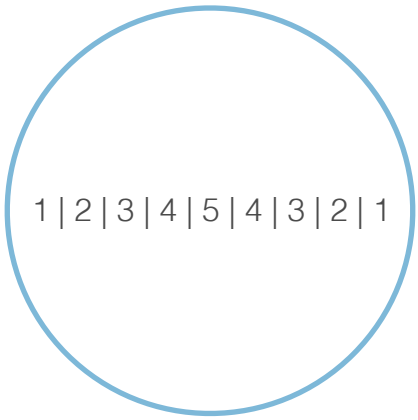
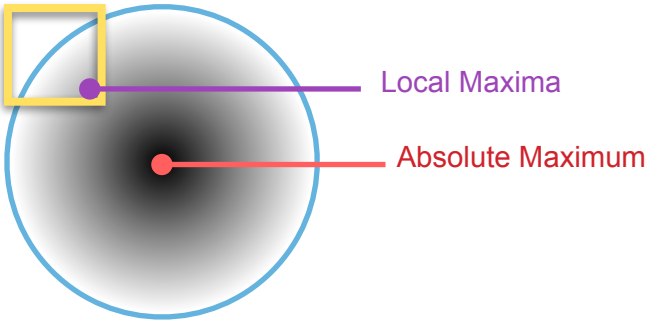


Figure 2.4: Distance transformation on a circle. Each number represents the distance transformation value for each ring of pixels that make up the circle of radius 5 pixels

The set of central points for a given blob is constructed by sweeping the distance transformation with a ’n * n’ mask. For each increment by ’n’ through the distance transformation with the mask, a local local maximum value is found inside the mask. If this local maximum is within 50% of the absolute maximum (the largest value in the distance transformation), its location and value are save in the set of central points for a

given blob. **Figure 2.5** below helps explain this by continuing with the distance transformation of a circle blob.

n * n mask.
Mask moves by n units to the right until it reaches the the far right of the image. It then resets to the left side, and is shifted down by n units. This effectively “sweeps” the distance transformation. In each increment by n, a local maxima is found, and kept if it is with in 50% of the absolute maxima.



Circular blob, distance transformation. Darker region is higher value. Border drawn for shape visibility.

Figure 2.5: Algorithm for finding central points on a circle shaped blob, which has the distance transformation applied to it. Transformation values increase from light to dark

Local maxima that are within 50% of the absolute maxima have their location and value saved in a “central point”, the central point is added to a set. The resulting set of central points provide multiple points for feature tracking in the instance of blob occlusion, when two or more blobs overlap, and the resulting blob only has a centroid for tracking. Sharma[1] talks extensively about why central points are a stable feature to track during blob occlusion.

2.2.2 Scoring Matches of Identifiers and Blobs

Having established blob and identifier metrics, it is now possible to create equations for determining what is a “good match” between any given blob and identifier. Each identifier is scored against every blob; these scores are then stored in a sorted array of best (lowest) score first, in each identifier. An individual score is a structure of three “sub-scores” which are computed with the following 3 functions respectively:

D(O, B) = Euclidian distance between the (x,y) coordinates of identifier O, & blob B (1)

A(O, B) = difference in area = [area(O) – area(B)] ÷ max[area(O), area(B)] (2)

V(O, B) = ratio of overlapping area = overlapArea(O,B) ÷ min[area(O), area(B)] (3)

Functions 1, 2, 3: Scoring functions between blob B, and identifier O. The area of an identifier or a blob is found by multiplying their respective dimensions (height * width).

These three score functions between blob B and identifier O; distance 'D', difference in area 'A', and ratio of overlapping area 'V' are used slightly differently in each of the 3 stages that comprise of the matching algorithm of identifiers to blobs. The matching continues to the next stage only if the set of blobs for a given frame is not completely matched with an identifier. Note, that the matching is not necessarily one to one, and at the end, there may be no "good matches" for an identifier or blob. Also note that the identifiers location and dimensions used for scoring are the prediction made by the Kalman filter as shown in **Figure 2.5** above. The three stages of the identifier to blob matching algorithm are as follows:

Matching Stage 1: In the first stage, a one-to-one matching between identifiers and blobs is attempted. Identifiers save the scores against every blob in a least to greatest sorted array. Each score in the array is computed with the three sub-score functions between blob B and identifier O from above. In essence, the best match for an identifier is a blob that is close, has nearly identical area, and has a lot of overlap.

In the following scoring **Functions 4, 5,** and **6**; tD, tA, and tV are the threshold constants for max distance, max area difference, and minimum overlapping ratio respectively. The function maxThresh*=(T, X) returns infinity if X > T or, returns X otherwise. The function minThresh returns infinity if X < T, or 1 otherwise.

$$S1 = maxThresh(tD, D(O,B)) * maxThresh(tA, abs[\ A(O,B)\ ]) * minThresh(tV, V(O,B)) \quad (4)$$

Function 4: Stage 1 score calculation

$$maxThresh(T, x) = \{\ infinity\ if\ x > T,\ \ x\ otherwise\ \} \quad (5)$$

Function 5: Maximum threshold, essentially invalidates a score

$$minThresh(L, x) = \{\ infinity\ if\ x < L,\ \ x\ otherwise\ \} \quad (6)$$

Function 6: Minimum threshold, essentially invalidates a score

Each identifier holds an array of its scores against every blob for several reasons. First, is so that scores are computed only once, and each successive stage does not require more computations. Second, is so that a one to one match can be established for an identifiers best valid score (meaning scores that are not infinite). Because of this second reason, an iterative process is defined to resolve “collisions” between two or more identifiers whose best (lowest) match scores are with the same blob. This iterative process is shown in **Figure 2.6** below.

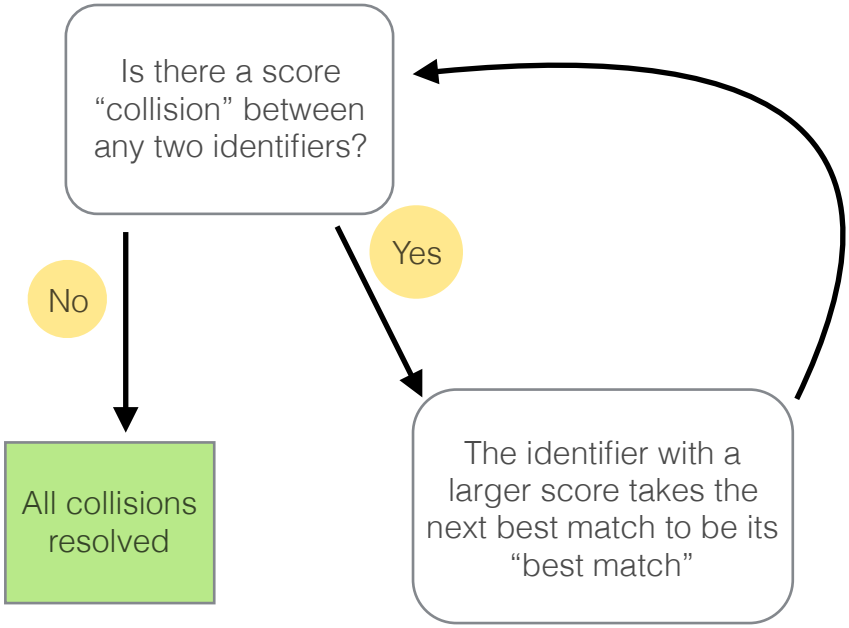


Figure 2.6: Iterative process for assuring one to one matching in Stage 1

After scoring identifiers against each blob, sorting the matches to find the best match, and resolving any match collisions, the result is a set of blobs and identifiers with one-to-one matching. The identifier saves the parameters of the blob that it is matched with. This information will be used later as the “measurement” information fed into the Kalman filter to obtain the Kalman filter estimated position, velocity, dimensions, and rate of dimensions change for the identifier, as shown in **Figure 2.3**. If there is any unmatched blob, the matching algorithm continues onto Stage 2.

Matching Stage 2: In the second stage, a one-to-many match is attempted between an unmatched identifier and unmatched blobs. An identifier tries to establish a match with many blobs who have a distance larger than tD, an overlap ratio greater than tV, and a smaller area, which results in A(O,B) to be positive as opposed to negative. **Function 7** shows the criteria needed for blobs to be considered for the one to many match.

S2 = minThresh(tV, V(O, B)) AND minThresh(tA, A(O,B)) AND minThresh(tD, D(O,B)) (7)

Function 7: Scoring conditions needed for one to many match

If two or more blobs satisfy these conditions for a given identifier, they are all combined to create a temporary “super blob.” The identifier is then matched with this “super blob” and the “super blobs” centroid and height and width to update the measurement data of its Kalman filter. Each blob used to create this super blob is considered matched. The matching algorithm continues to the 3rd and final stage if there are any unmatched blobs.

Matching Stage 3: In the third and final stage, the remaining unmatched identifiers and blobs attempt a many-to-one match. In this stage, each remaining unmatched blob has it's central points computed. Every unmatched identifier has a stage 3 match score computed against every central point. Basically, central points replace centroids and the best match is a blob whose central point is the shortest distance from an identifier.
If an identifier has not been matched in stage 3 in the previous stage, **Function 8** shows the Euclidian distance between an identifier's location and a blob's central point, where tDc is the threshold for a maximum allowed Euclidean distance. However, if an identifier has been matched in stage 3 in the previous frame, **Function 9** is used to calculate the score with the central point's distance transformation value squared.

S3a = maxThresh(tDc, D(O,B-C)) (8)

Function 8: The Euclidean distance between an identifier and a central point

S3b = maxThresh(tDc, D(O,B-C)) * maxThresh(tAc, abs[A(O,B-C)]) (9)

Function 9: Area function between an identifier and a central point

An identifier is matched with a blob whose central point produces the lowest score. The blob and identifier are marked as matched. Now the Kalman filter must be updated with the appropriate measurement values, as was done for the prior two stages. The location is fairly straight-forward. The central points location is used as Kalman filter measurement data, and the identifiers location is updated with the Kalman filter's estimated location.

Dimensions on the other hand are not as straight forward. Sharma[1] does not say how the central point value is used to update the identifier's dimensions. Because of **Function 9**, it can be assumed that the central point value must be used to update the

blob dimensions in some way. However, central point values are a single dimension and an identifier has two dimensions: height and width. The simple solution is to use the central point value as height and width measurement data. The shortcomings of this are discussed in greater detail farther down below as they have a fairly large impact on the algorithm.

2.2.3 Identifier Stability Improvements

A contribution this paper makes to the work done by Sharma[1] is to improve the stability of identifiers. Because it takes some time for a Kalman filter to stabilize its predictions, and matches are based on hard thresholds, there will come a time when a blob is perpetually be matched only for a single frame. Each frame a new identifier is created for the blob because it failed to be matched with the identifier created in the previous frame. Since identifiers continue with a constant velocity for several frames until they “die,” it is possible for them to conflict with other identifiers ability to match to their corresponding blob. Other times, an identifier may miss its target for a frame or two, perhaps due to occlusions or other reasons, but be matched up with its blob afterwards.

To account for this, two layers of identifiers are used: a stable set and an unstable set. All new identifiers start off in the unstable set. After several successive successful matches, an identifier is promoted to the stable set. Once in the stable set, an identifier is demoted if it goes unmatched for several successive frames. The stable identifier set goes through all three of the aforementioned matching stages. The left over set of unmatched blobs is then passed to the unmatched identifier set for matching using the same three stages.

Having two identifier sets is quite beneficial because it relaxes the requirements of the Kalman filter prediction. Previously, if a Kalman filter produced a bad prediction for

just a single frame, the blob it was matched with would be replaced by a new identifier, with a newly instantiated Kalman filter. This is undesirable because Kalman filters only perform well after several iterations of measurement data. It is quite difficult to create a Kalman filter that produces accurate predictions within its first iteration. Additionally, an identifier is essentially allowed several frames of missed matches before a new identifier takes it's place. This policy promotes older identifiers who have accumulated good predictive information about a particular blob and grants the identifier a few mistakes.

2.3 System Stage 3: Dynamic Line Model

A probability image is used to describe the probability that a given pixel corresponds to the region a line forms in the video scene. Then the location of every stable identifier is plotted on this probability image to determine the probability of being in line or not. Then the number of people in line, along with a confidence interval, is reported by the system.

During the development process, a static line model was used. This is a hand generated probability model of where a human thinks a line forms in the scene. However, this technique does not allow for the line to organically shrink or grow. This is a big limitation that will produce many false positives. Essentially, any person that walks into this image will be considered part of the line. Another key contribution of this paper is for a technique to dynamically determine a line region in a scene.

The base assumption about a line is that people are standing still. To capture this, two ABS are used. The first ABS is very adaptive, so that it does not take long for an individual standing still to fade into the background. The background model of this ABS contains every individual standing still. This background image is then passed into an extremely slowly adaptive ABS in order to extract the people standing still. This

extremely slowly adaptive ABS is set to capture the “true” background model. More importantly though, this second ABS produces a foreground binary image that captures only the people standing still, but only standing still for a certain amount of time.

This is sort of a temporal bandpass filter for motion. People walking past the line are not picked up. People who stand still are captured as foreground objects, and noise in the form of drastic changes to the background are eventually captured. To the best of this author’s knowledge, no other paper has used ABS in such a manner. (Side Note: Besides its use in generating a dynamic line probability model, this targeting technique is also used as an alternative to the regular blob detection technique presented earlier in this paper.)

There are two additional steps that need to be taken before the binary image of standing people can be used. First, a large gaussian filter is used to blur the binary image. This gives the probability image soft edges and variable levels of certainty. The last step is to perform a histogram equalization on the image. This is because we want a full probability range from zero to one, and the gaussian blur can cause regions of the probability image to loose their maximum value. **Figure 2.7** below shows the full system block diagram.

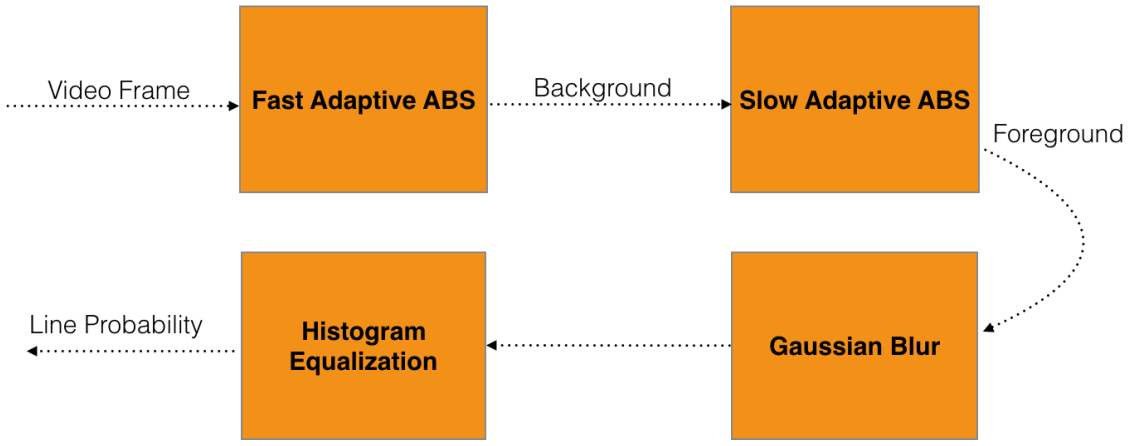


Figure 2.7: Block diagram of how a dynamic line model is generated

3. Implementation

The core of this project is implemented using the extensive computer vision library provided by OpenCV and the great performance and parallelism provided by C++11. Additional libraries are used for the interface with the MySQL database. The system itself has a command line interface, and can be configured using a configuration file. A testing system is written in python to allow multiple instances of the system to be run at once. The testing system and configuration input file enable for autonomous system optimization. With over 25 different parameters, the search space for the optimal configuration is rather large. Performance is necessary in order to meet the real-time requirement, and to make searching the system configuration space feasible.

Multithreading is used to achieve better performance. Because of the stateful nature, and sequential dependencies in certain system stages, a pipeline design is used. Each stage of the system (target acquisition, target tracking, and line probability model) is placed on a separate thread, all managed by the main thread which also handles video I/O. A protected queue is used to pass information from one stage to the next. The queue has a simple interface shown in **Figure 3.1.** Each operation is tightly protected by a mutex semaphore.

```
ProtectedQueue<Type>(int maxSize)
Type pop()
void push(Type obj)
bool empty()
```

Figure 3.1: Protected Queue Interface

Proper multithread design is necessary to not only assure performance benefits, but also to reduce the change of multithread related bugs from creeping in. The pipeline design, along with the protected queues offer this by ensuring a proper, and secure data path. **Figure 3.2** shows the systems multithread design. The Display Frames Queue is necessary to ensure that only the main thread interfaces with OpenCV's video I/O operations. This is because the GUI used to display different parts of the system is on a separate thread due to the underlying implementation of OpenCV. The only way to prevent race conditions is to have only one thread handle video I/O. Besides the Display Frame Queue, the design follows pretty closely to the pipeline design commonly used in hardware for modern processors.

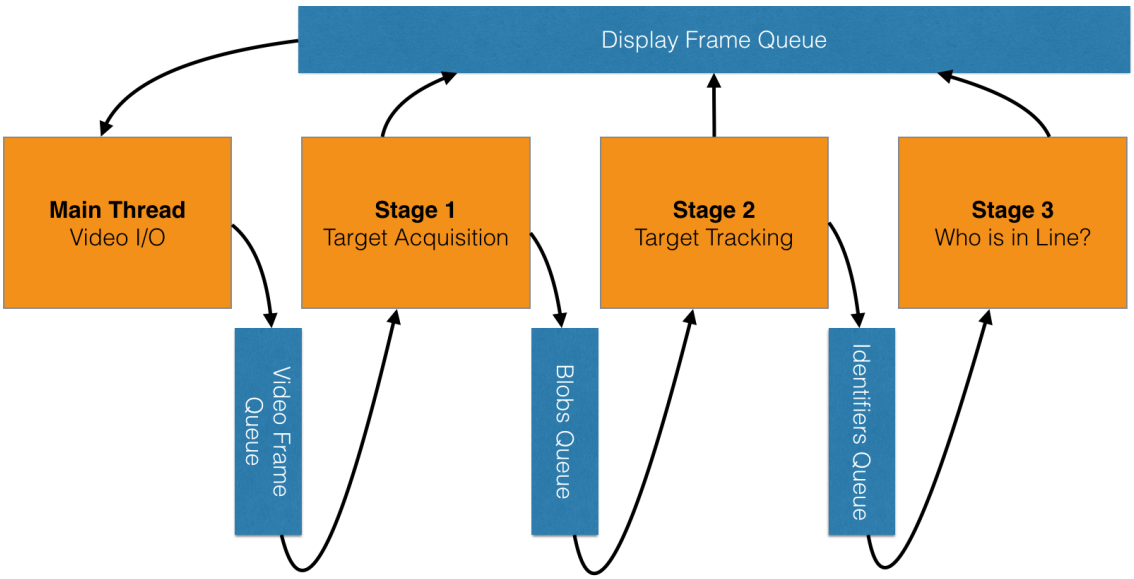


Figure 3.2: Multithreaded Pipeline Design

4. Challenges and Limitations of the System

While many papers[1,6, 9, 10] are quick to highlight their achievements, they often fail to mention their limitations. Highly controlled scenarios, while important for generating favorable results, often fail to capture performance in less favorable conditions. To truly understand modern computer vision systems, it is important to look at failures just as much as successes. It is only through a deep examination of challenging video sequences that fundamental limitations of modern systems can be understood. Only after understanding shortcomings can new and better techniques be developed. Therefore, this paper will present some of the challenges faced by this computer vision system.

The influence of real world events and practical constraints presents a set of challenges, typically in the form of noise that must be overcome in order to obtain accurate results. Because of the sequential nature of the system, noise compounds from each stage, and influences the final count of people in line. To a certain degree, each of the follow sources of noise is orthogonal, meaning they are independent of one another, and reducing the effects of one noise source does not guarantee reduction in the effects of another. In each stage of the system, the following noises can be attributed to a reduction in accuracy:

1. Camera Position
2. Lighting
3. Homogenous Mixture of Stationary and Moving Targets
4. Extreme Occlusion

Figure 4.1: Example of the four noise sources. The green circle is stationary people in line. These targets have a perspective change due to camera position, they are overexposed due to lighting coming from the upper left corner, and they are subject to extreme occlusion. The red circle is people in motion. These people walk in front of the stationary targets. not only causing further occlusion, but also challenges for any background subtraction techniques

First, the effects of these noise sources will be examined for the target acquisition stage of this system, followed by target tracking stage, and finally the application specific stage. Mitigation of these noise sources for system performance analysis falls out of scope of this paper.

4.1 Target Acquisition Limitations

Adverse effects from the aforementioned noise sources are perhaps most influential in target acquisition. With bad targets, or partially captured targets, it makes it even more difficult to track and account for occlusion with the techniques discussed in section **2.2 Target Tracking**.

4.1.1 Camera Position Challenges in Target Acquisition

The ideal camera position would be orthogonal and in central to a line of people, as to capture the space between each member of the line. Any deviation from this would introduce additional occlusions, or sizing differences between targets due the perspective. Additionally, the camera should be far enough away from the line so that a narrow angle lens can be used to capture the entire line in high fidelity. Wide-angle lenses would need to be closer to the line to capture the line in adequate fidelity, and being closer would introduce occlusion at the beginning and end of the line of people.

4.1.2 Lighting Challenges in Target Acquisition

The ideal lighting scenario would be fully ambient lighting. This would prevent the formation of shadows, sunspots, and overexposure.

Shadows are challenging for ABS. Despite research[3,4] in shadow resistant ABS techniques, which usually check for darker shades of the background model, it is still very challenging to differentiate between a true shadow and darker colored clothing. Also, shadows become out of focus, the further away an object is from the shadow-projected plane. This effect is amplified in reflective tile floors.

Sunspots are also challenging for ABS, especially very bright spots on darker backgrounds. The deviation from the background model is significant enough for the sunspot to be interpreted as part of the foreground. Besides causing false positive foreground pixels sun spots, or more specifically the source of the sunspot, can be a source of extreme lighting dynamics. These extreme dynamics cause overexposure in some parts of the image or underexposure in others. Improperly exposed images could appear desaturated, making the color differences between the background model and

foreground objects smaller. This all leads to false classification of foreground or background pixels.

Figure 4.2: Video frame of a very busy line, which presents great challenges for target acquisition due to occlusion, lighting, mixed motion, and camera positioning

It can be seen in **Figure 4.2** above that the used video for this system has a lot of occlusion in the front of the line due to the camera angle. Even a human has to thoroughly examine dense clusters of people to accurately hand count the number of people in line. In fact it is the opinion of the author that virtually any targeting algorithm currently available (either in literature or production) will have targeting difficulties with the amount of occlusion present.

Continuing the examination of the scene in **Figure 4.2,** the upper left corner is overexposed due to the bright lighting caused by the sun shining through the sliding door. This causes a silhouetting effect for people in the front of the line, making them appear desaturated and similar in color. The extreme light dynamics furthermore are the source of underexposure in other parts of the image, yet again resulting in desaturation and a more homogenous color profile. Additionally the sliding door in the upper left corner causes sunspots in the lower right corner of the scene when open. It also does not help that the door is one of the main entrances for the building and is utilized very frequently.

Figure 4.3: An extremely challenging scene for human and computer vision target acquisition systems. There are 10 people in this frame

Poor camera positioning and lighting can be tremendously challenging even for a human. As a challenge, examine **Figure 4.3** above. If the human visual system is having trouble correctly counting the number of people in line an algorithm will surely struggle

as well. Scenes like this can often arise throughout the day and will typically cause severe undercounting.

4.1.3 Mixed Motion Challenges in Target Acquisition

The ideal motion of the video scene for ABS would be constant for all targets needing detection. This would allow the ABS to work under the designed presumption that foreground objects are inherently moving, and background object are inherently stationary. Therefore having a homogenous mixture of both moving and stationary targets is fundamentally challenging for ABS. Foreground objects that stand still for long enough faded into the background and can be considered noise in the background model. Not only are they not counted as foreground pixels, but the noise could also cause additional foreground objects of similar color to be partially falsely classified as part of the background.

Another fundamentally important fact of ABS is that the majority of temporal samples per pixel are part of the background model, and foreground pixels are merely outliers or impulses to the norm. However, real world scenarios do not offer this. During coffee rush hour, the line can constantly be 10+ people long for several hours. At specific parts of the scene, a true background pixel might not be obtained for the length of this time. Despite the tunable adaptivity parameter in many ABS techniques, going to the extremely slow learning spectrum of adaptivity would undermine key advantages of ABS.

The noises caused by mixed motion are unfortunately present in the video used for this system. People waiting in line stand still, fade into the background and become impossible to target. Even the dual ABS technique, which targets stationary people is susceptible to noise caused by mixed motion. When people move forwards in line, they

are not targeted because they are in motion and not standing still. The effect of this noise source is partially captured blobs of improperly classified foreground pixels.

4.2 Target Tracking Limitations

All of the aforementioned challenges in target acquisition carry over in target tracking, because target tracking uses data acquired in the target acquisition stage. The position and target dimension predictions made by the Kalman filter become less accurate, and therefore the matching algorithm of identifiers to blobs makes false matches. That being said, noise from target acquisition is not the only challenge for the tracking stage of the algorithm.

4.2.1 Kalman Filter Challenges in Target Tracking

In the ideal scenario, blob shapes would be rigid, meaning every blob shape is constant. However, this will never be the case because of the way people walk, the size difference caused by perspective, and even the micro motions that occur when standing still. Even if target acquisition produced 100% accurate blob data, the blob centroid or central points used as Kalman filter measurement data could shift slightly because of the changes in blob shape. These slight shifts become measurement noise for the Kalman filter.

To further understand this, imagine both a human and a big ball moving at a constant velocity in a scene. The ball will have a rigid blob shape and its centroid will truly be the center of the ball, and therefore the measurement data will not have noise when being fed into the Kalman filter. The human on the other hand will be moving their arms and legs when walking, altering their blob shape in each frame. This causes the

blob centroid to not accurately represent the center of the human, and therefore counts as noise in the Kalman filter measurement data.

Another ideal scenario would be to have a tracked target's motion be non-chaotic and completely described by the second order model used by the Kalman filter. People would not make sporadic direction changes, speed changes or suddenly come to a stop. These chaotic behaviors are problematic for the Kalman filter and demand multiple iterations for the prediction accuracy to improve.

4.2.2 Matching Challenges in Target Tracking

All these inaccuracies; noisy target acquisition, chaotic target motion, noisy centroid positions, and Kalman filter convergence times, all contribute to a certain degree to a noisy Kalman filter prediction, which in turn leads to potentially bad matches.

The most common case of this is a failure to establish a match. In such a case, a new identifier is created for any blob that has not been matched. This means a new Kalman filter is created for both the target position and dimensions. Normally this would not be a problem, but the Kalman filter demands several iterations to produce accurate predictions and stabilize. Prior to this, the Kalman filter is unstable and the new identifier could fail to be matched in the next iteration. While the stable and unstable set of identifiers proposed in this paper aim to alleviate this problem, it could still occur, albeit less often.

Another issue with creating new identifiers for unmatched blobs is that the algorithm always assumes a blob is a single person. This can happen if the occlusion matching fails to produce a good match on a blob of two or more people, and a new identifier is created for the blob. However the more frequently observed case is when tightly clustered group of people walk into the scene. The group is counted as one

person, and unless the blob splits, it will continue to be counted. The occlusion resistant technique only works when the merged conglomerate blob consists of two or more well tracked blobs beforehand.

A less common misclassification can occur during the second stage of matching. Sometimes two nearby blobs fail to be matched in the first stage (one to one matching) and proceed to the second stage of matching where the blobs are falsely assumed to be split. This causes undercounting because two (or more) people are counted as one. While the idea behind this stage is good, because sometimes a blob of an individual can split into two or more blobs for whatever reason, it can cause issues as in this case.

The last issue encountered in the matching algorithm occurs in the third stage of the matching algorithm, where multiple identifiers match to a single blob in order to deal with occlusion. In the ideal case, only identifiers who's blobs have merged into one should ever reach this stage. Unfortunately due to noisy Kalman filter predictions, stray identifiers looking for a match, and tunable match parameters, sometimes identifiers will falsely double count a single person by falsely being matched in the third stage. This is a challenging problem to fix because while visually it may be obvious that a blob belongs to just a single person, from the algorithms perspective, it has no way of knowing. A more robust algorithm is needed, one that leverages a plethora of different features, to truly alleviate this problem.

4.3 Line Probability Model Challenges

Two key features of the line probability model is that it relies on people standing still, and leverages no priori knowledge about where a line forms. While these features provide robustness and a general approach for a wide range of environments without

requiring training data for the algorithm, it can also be a shortcoming and challenge in its own right.

The first challenge is capturing a probability model for only the line to place an order. Often times after placing an order, a person lingers around nearby the line they were just in, waiting to receive their order. As a result, these people stand still and are counted as part of the line probability model. This can cause false positives, and generally a larger line region.

Another challenge for the line probability model is the time it takes for a person to be captured as part of the background in the first ABS. The time it takes to acquire a target can contribute to undercounting, since a smaller line region will be observed in the meantime. Unfortunately, there is nothing that can be done to alleviate this. A careful balance must be achieved such that the dual ABS technique used to generate the line probability model does not capture slow walking pedestrians, and only capture people who have truly stopped and committed to standing still in line. Variable video FPS only make this problem harder since the ABS technique is frame based, and not time based.

This challenge also presents another problem. When the line advances forward, and everyone moves, there is a moment where every person in line is moving, and will therefore not be standing still. This means that they may not be counted in the line probability model.

5. Results

In order to test the computer vision system, a Python script test platform has been written to automate the process. This script runs the computer vision system with a specific configuration file, and test video. It saves the system outputs and the input video frames per second to an output file to later be compared with a hand generated expected output file. This creates a simple environment for creating multiple tests, since all that is needed is system inputs and expected outputs to compare against. Another nice feature of the testing platform is its ability to try multiple configurations files for the same test. This enables optimization of the system by intelligent traversal of the system configuration space. However this work is out of the scope of the paper.

In order to compare system outputs to the expected outputs, discrete time signals must be created of both the expected and actual outputs. The actual output as produced by the computer vision system saves the video FPS at the top of the output file, followed by the line size at each frame. Having the video FPS is important since it is essentially the sampling rate for the discrete time signal of the actual output. The sampling rate is used to generate the expected output signal from the human recorded pairs of time and line size. For both actual and expected signals, step interpolation is used.

After both the expected and actual signals are generated, the Root Mean Squared Error (RMSE), as seen in **Function 7**, is computed between the two signals. The RMSE shows, give or take, the number of people miscounted. Low RMSE values are favorable because it means the system is pretty close to the truth. Also statistics are performed on the error signal, which is the difference between actual and expected. The mean and standard deviation of the error are calculated in order to see if the system is under counting, or over counting.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y)^2}{n}}. \quad (7)$$

Function 7: Root Mean Squared Error between the actual (y_t) and expected (y)

There are two targeting algorithms being tested, with the key difference being the target acquisition technique being used. The first algorithm uses the traditional ABS technique for target acquisition. The second algorithm uses the DABS technique presented in this paper for the same task. It is important to remember that both algorithms also use the DABS technique in the dynamic line model stage, since the DABS technique was first developed for this purpose.

5.1 Test Videos

Three videos are used for testing three different scenarios faced by the system: a short, medium, and long line. A prerequisite for any test video is that its first several frames should have no people in them, and no line for about one minute. This allows for the ABS and DABS to create a noise free background model. Without a good background model, the performance of the whole system suffers.

The short line video is 11 minutes long, and has line sizes that span from zero to five people. It is recorded in the evening. Typically during this time there are (understandably) not very many people who wish to get coffee. Despite the short lines and relatively few people, some interesting challenges appear in this video. The slightly dimmer lighting causes colors to desaturate, and the relatively fast moving lines can be a little too fast for the DABS technique in the dynamic line model stage. Several frames of the video can be seen in **Figure 5.1**.

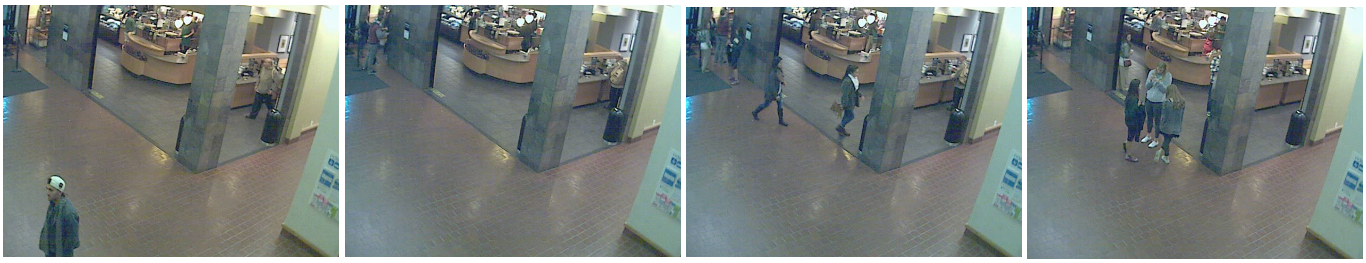


Figure 5.1: Several frames from the short line video

The medium line video is 14 minutes in length and has between zero to nine people. This video is recorded in the afternoon. Due to the shorter days, the video experiences similar challenges as the short line video. Color desaturation due to poor lighting causes some interesting challenges. Also this video has a large group of people clustered at the front of the line which causes extreme occlusion. Several frames of the video can be seen in **Figure 5.2**.

Figure 5.2: Several frames from the medium line video

The long line video is 15 minutes in length and peaks at a line length of 18 people. The video was recorded in the morning when the line for coffee was on average 10 people long. This is probably the most challenging of the videos to process. The video has a lot of people walking past the line, challenging lighting in the front of the front of the line, and ample occlusion. Several frames of the video can be seen in **Figure 5.3**.

Figure 5.3: Several frames from the long line video

5.2 Test Results

The RMSE of each test can be seen in **Figure 5.4**. This provides a good overview of performance by showing the relative miscount of the system with units being people. The test results for each test video and targeting algorithm can be seen in **Figure 5.5** below. For these results, the first 1500 frames of output have been discarded because it takes a few minutes for the ABS and DABS to create good background models. As stated before, if a poor background model is used poor results are to be expected. The error distribution for each test result is presented in **Figure 5.6**. Lastly, **Figure 5.7** shows the mean and standard deviation for these distributions.

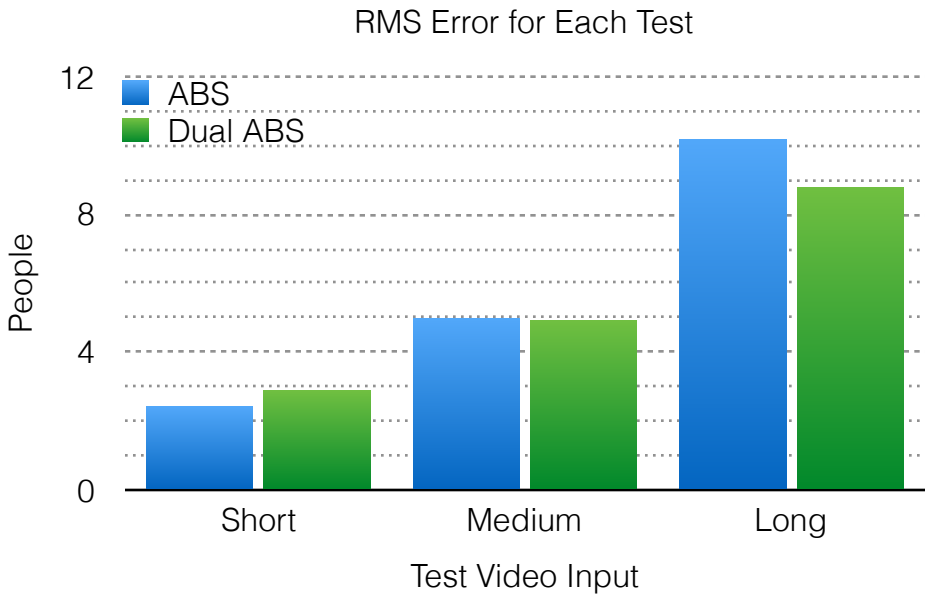


Figure 5.4: RMS Error for each test, for both ABS targeting and DABS targeting algorithms, ran on the three videos

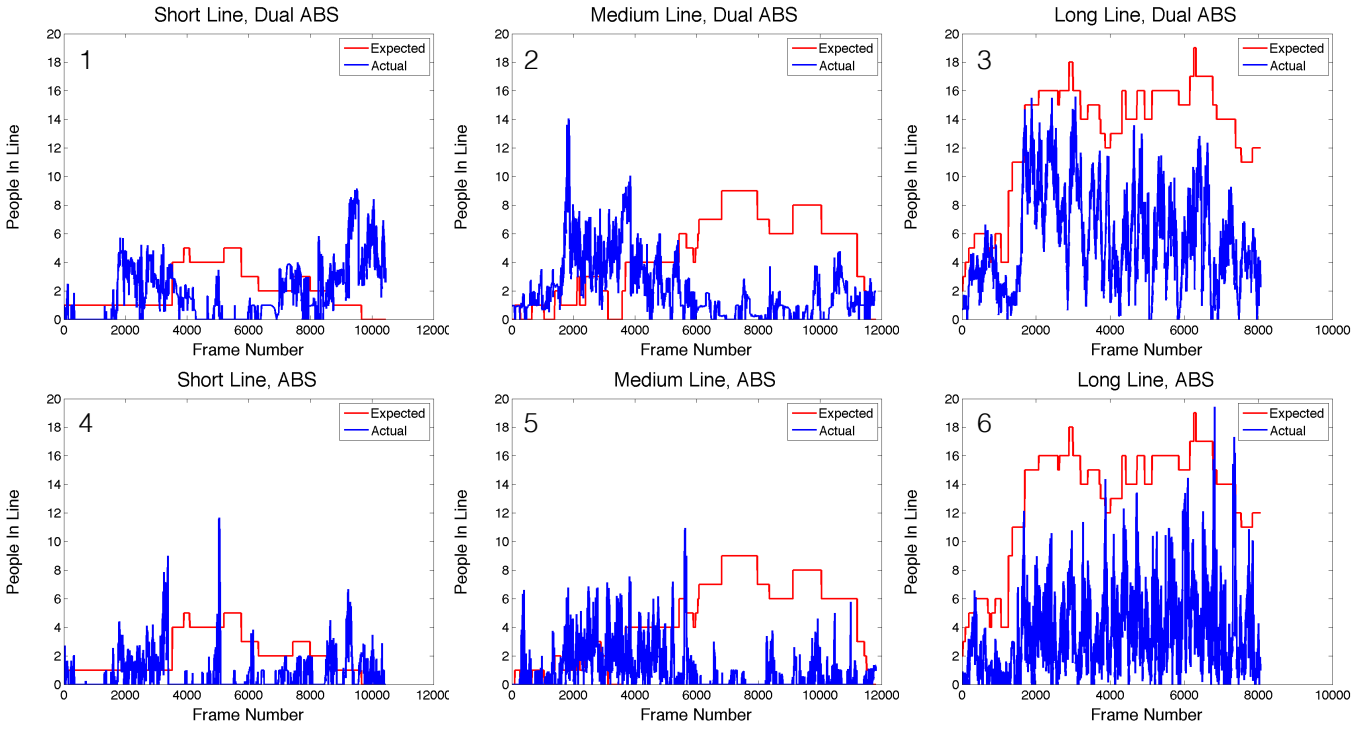


Figure 5.5: Test results for both ABS and DABS targeting algorithms, tested on the three test videos. The first 1500 frames are discarded to allow for a proper background model to develop. Figures 1 to 3 use dual adaptive background subtraction for targeting, while Figures 3 to 6 use regular adaptive background subtraction for targeting

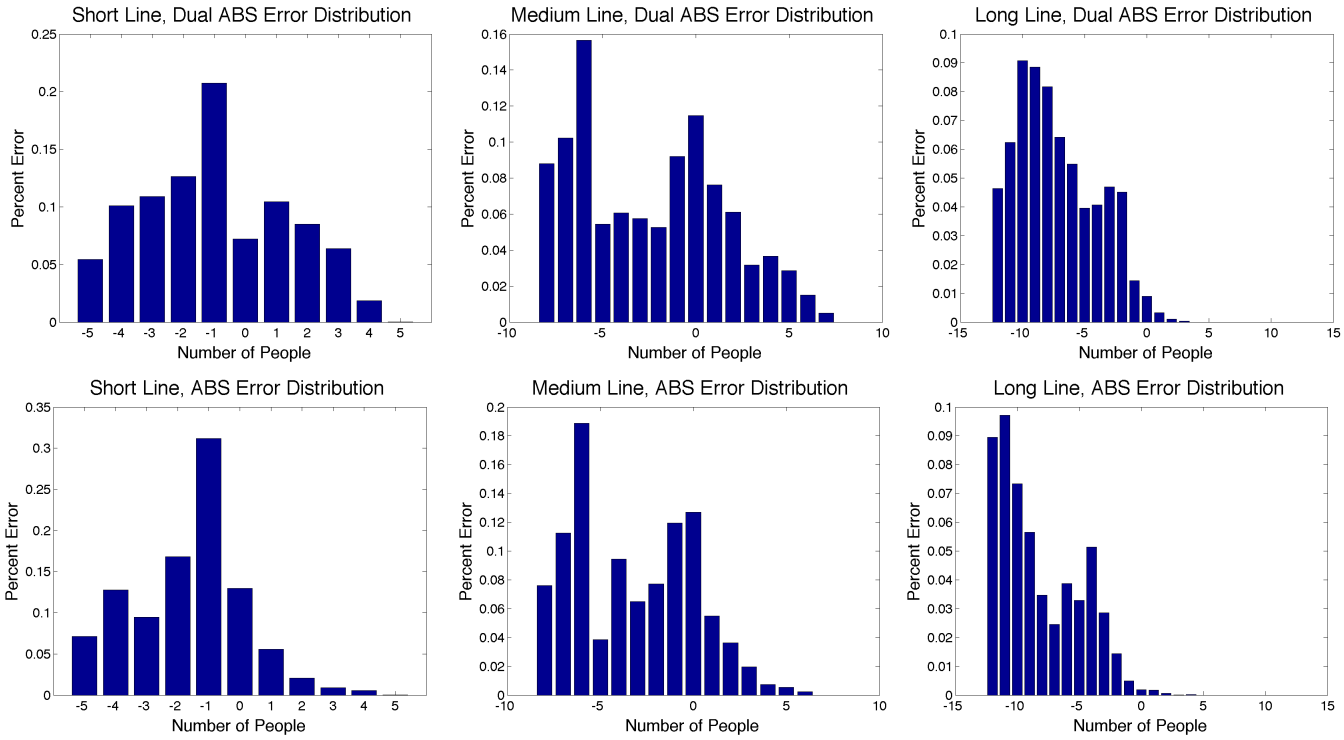


Figure 5.6: Statistical distribution of the error

Distribution	Short Line, ABS	Short Line, DABS	Medium Line, ABS	Medium Line, DABS	Long Line, ABS	Long Line, DABS
Mean	-1.3513	-0.15281	-3.471	-2.3629	-9.4376	-7.4494
Standard Deviation	2.041	2.8909	3.626	4.3531	3.7898	3.7471

Figure 5.7: Mean and standard deviations for error distributions

5.3 Results Discussion

There are times when the system performed reasonably well, and produced outputs close to the expected. The system using DABS for targeting was very much on the mark till about frame 3500 of **Figure 5.5.3**, the long line test video. The system using ABS for targeting seemed to do well in **Figure 5.5.5** till about frame 5500 of the medium line test video. It can be seen from the RMSE results in **Figure 5.4** that DABS generally performs better than the ABS. For the long line test video, DABS performed 20% better than ABS. Additionally, throughout all the error distributions, DABS had a smaller mean error, albeit slightly larger standard deviation.

In general, however, both versions of the system seem to undercount more often than over count. This could be attributed to the stability improvements to the identifier that prevent stray, unmatched, identifiers from causing double counting. Also, this could be attributed to the dynamic line model, which reduces the number of identifier people in the scene to only those that are in line. However with the omission of the dynamic line model, over counting was a bigger issue with earlier iterations of the system than undercounting.

While results do appear to be a little lackluster, the test videos used, and the real world environment produce some very challenging scenarios. For a more in depth examination of all the challenges faced by this system, refer back to Section **4. Challenges and Limitations of the System**. Challenging scenarios can also occur to humans, were we have to stop and ask every once in a while if a person is in line or not. Perhaps the design criterion of maximum system autonomy is a bit debilitating. Having as much memory and knowledge about what defines a line could be beneficial to this computer vision system. For example if a user is allowed to give the system priori knowledge about where a line starts, in this case the upper left corner of the video, the over counting observed in **Figure 5.5.1** after frame 8000 could be avoided because the system picks up people who are standing still after placing their order, or perhaps just stopping to have a conversation in from of the coffee shop. **Figure 5.8** below shows the cause of the over counting occurring in **Figure 5.5.1**.

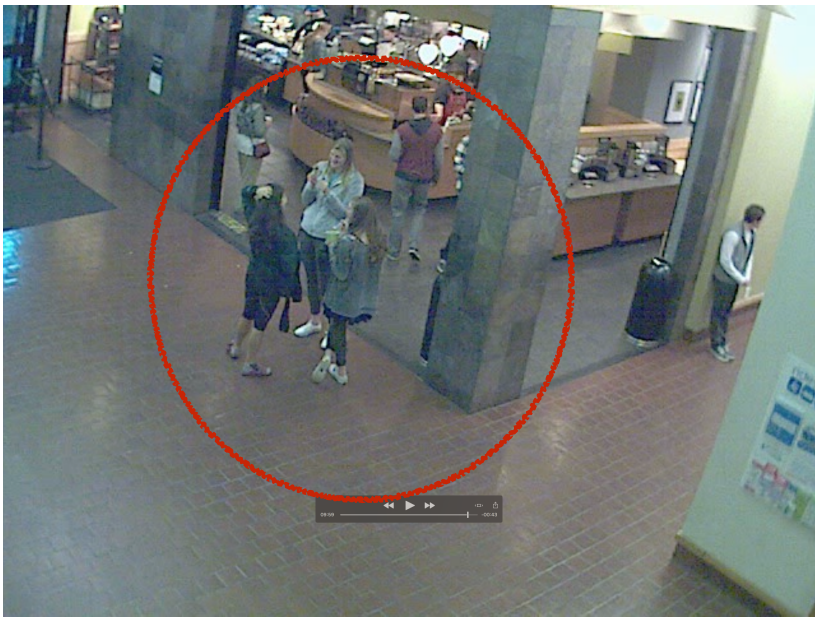


Figure 5.8: People standing around can be falsely counted as standing in line

However, not all miscounts or errors in the actual results have an easy solution. The severe undercounting that occurs after frame 6000 in the medium line video (**Figure 5.5.2** and **Figure 5.5.5**) is caused by an extremely challenging video sequence. Poor lighting, extreme occlusion, similar colored clothing as the background, and even video noise contribute to the system failing to register the cluster of people shown in **Figure 5.9**. Even a hand count is rather difficult, try one with the figure, and for the actual value refer to **Figure 4.3**.

Lastly, the fluctuations in the results of the long line video (**Figure 5.5.3** and **Figure 5.5.6**) can be attributed to multiple points of occlusion, which can be seen in the last two frames of the long line video sequence in **Figure 5.3**. In the case of the system using ABS for targeting, these fluctuations can also be attributed to people standing still and fading into the background. Although the DABS targeting algorithm set out to remedy this problem by targeting people standing still, there are brief moments when the line moves forwards that all acquired targets disappear because they all take a few steps forward in line. This is part of the reason as to why the DABS targeting algorithm experiences fluctuations, although they are lower in frequency than those of the ABS targeting algorithm.

Figure 5.9: The tight cluster of similarly colored pedestrians is responsible for the severe under counting seen in the medium line video results

6. Related Works

There are many different proposed systems available to track pedestrians in a video, each with their inherent advantages and disadvantages. Typically tracking is abstracted into it's own level and contains a set of identifiers which correspond to pedestrians found in a scene. Tracking systems will usually have a mechanism for target acquisition, matching algorithm(s) based on extracted features that try to match an identifier to the same person in future frames. More sophisticated systems very often have stateful identifiers capable of making predictions of sorts to improve accuracy and reduce search spaces.

6.1 Target Acquisition Related Works

Target acquisition is typically the first step of finding pedestrians in a scene. This can be done by hand but many autonomous techniques are available, each leveraging different features. One popular technique is Histogram of Gradients (HoG) detection [5] which uses a known visual gradient model of a pedestrian and searches for it in the scene. This is in contrast to DABS targeting which relies on specific motion characteristics instead of a visual model. However the performance of HoG detection relies heavily on the gradient model used and performs poorly when camera angles, pedestrian shape and size, or occlusion cause deviations from this model.

Adaptive background subtraction[1][3][4] is another popular technique for target acquisition. This is one of the technique chosen in this paper because of it's strengths in acquiring targets of any shape or size. It produces foreground pixels from objects in motion that are later converted into blobs through connected component analysis. One downfall is that pedestrians can be 'absorbed' into the background model if they are

stationary for extended periods of time. The DABS targeting technique presented in this paper is less susceptible to this because it specifically targets stationary people.

Regardless of how a target is acquired, additional features may be extracted from the target to form a model of the particular pedestrian. These features will aid the frame to frame tracking done by various matching algorithms. Features are typically chosen based on various considerations such as accuracy, computational complexity, and camera environment. Features may be classified as holistic (e.g. directly identifying a pedestrian through HoG or ABS), a decomposition of a pedestrian (e.g. identifying sub-parts that make up one pedestrian), or a mixture of the two (e.g. identifying a region of interest that could directly be a pedestrian, but extract additional features from the identified region in the case of occlusion). The DABS targeting technique is holistic since it aims to directly identify pedestrians, although the central points used for occlusion resilience in the tracking stage of the system could be considered a decomposition. New features are being proposed regularly, so it can be difficult to find the “best” feature or set of features, especially when considering how the features are used for matching.

6.2 Target Tracking Related Works

Tracking happens at every frame with the goal of tracking identified pedestrian in the previous frame with an identified pedestrian in subsequent frames based on specific features or models. Many different approaches exist, each technique somewhat dependent on the underlying features: color, blobs, or HoG, for example. One popular color based tracking is Continuously Adaptive Mean Shift (Camshift)[8], which is based on the Mean Shift technique[7] and iterative searches.

Typically novel tracking techniques, such as Camshift, are developed for tracking a single target only. To provide the tracking system noise free features to track, a human hand selects the target. Autonomously selected targets, like the targets produced by the DABS targeting technique presented in this paper, can have noise which can degrade a tracker's performance. Matching systems for multiple objects are inherently more complex than those that track individual objects. Additional considerations have to be made about how to resolve collisions in matches. The tracking system used in this paper is capable of tracking multiple targets based on noisy target data.

More sophisticated Camshift based trackers have also been devised based on target decomposition[6]. This gives multiple points to track a single target which can be helpful in improving the trackers accuracy, especially in the presence of occlusion. While the test videos in this paper have ample occlusion, trackers such as this are not viable because they cannot track multiple targets.

Typically, tracking systems will typically have a prediction mechanism or system to improve accuracy or reduce search spaces. For example, Kalman filters or particle filters are commonly used to provide a prediction of where a pedestrian moved in a new frame. Having some knowledge about where a target might be in the next frame can be crucial for reducing the Camshift or HoG detection search space[9,11]. The blob based tracker used in this paper benefits from Kalman filters since their prediction is the basis of finding matches between blobs. The predictive element of a tracking system does not have to be limited to position predictions. Adaptive appearance based tracking[2] have also been proposed to predict the appearance of a target. However these prediction systems require precise calibration for their given environment or target and may underperform otherwise. Also they may require a few good samples to "lock-on" to a

particular target. Should a target suddenly change directions or appearance, these prediction systems can take multiple samples to recover, if at all.

7. Future Work

There are many things that can be done to improve the computer vision system presented in this paper. For starters it would be interesting to run an optimization algorithm with the testing platform to find a more optimal configuration for the computer vision system. In order to make this effective for the live video stream where the system is ultimately designed to run on, the webcam should be configured, if possible, to run at a constant frame rate. The test videos recorded from the webcam should also be the same frame rate because several techniques used in the system have their configurations depended on the sample rate. This would be an immediate way to improve the computer vision system without changing the algorithms and techniques inside it. Tangental to this, it would be nice to the the whole line counting system fulling running and usable by students. This means the website, server, and database need to be up and running 24/7. On the server, a watchdog script needs to be created to ensure the computer vision system, MySQL database, and node.js backend can properly recover from a crash, or boot up if the servers go down for maintenance.

In terms of other algorithms to try, it is the authors belief that most of the work should be focused on target acquisition and to a lesser extent, tracking. The dynamic line model seems to perform decently enough and there are not many other techniques out there that can produce this kind of data. On the other hand, target tracking and acquisition are well established computer vision problems with many new techniques being developed all the time.

For target acquisition it could be interesting to employ some sort of neural network or scale invariant HoG detection. This could be run in conjunction with the output of the first ABS of DABS targeting proposed in this paper. The adaptive background model of the fast ABS seems to capture perfectly people that walk in line

and stand still. Currently an extremely slowly adaptive ABS is used to process this background frame to acquire targets. However instead of this, a neural network trained in pedestrian detection, or a scale invariant HoG detection technique could produce very interesting results.

For target tracking, the color profile of the blob could be used for Camshift, in conjunction with Kalman filter predictions for a reduced search space. The color profile of the target could be stored in the identifier (C++) class, which is currently created to store information about the blob being tracked. However for Camshift to be viable, several key problems must be overcome. First is identifier collisions, meaning two or more identifiers using Camshift have found a match within a very similar region. This needs to be addressed because if a person walks out of the frame, it would be desirable for the person's identifier, which would be storing the color profile used for Camshift, to die in order to prevent double counting. The second thing that needs addressing is having a group of people walk in the frame at the same time. If the group is tightly clustered, each person wouldn't have their own blob which would be sampled for color information, but instead it would be one large conglomerate blob which would be counted as one person. The only way to address this would be to use a different target acquisition method.

8. Conclusion

In this paper, a computer vision system for counting people standing in line has been presented. The system autonomously detects targets, tracks them, and determines how many people are in line. Through test results, it was shown that the novel DABS technique outperforms regular ABS for target acquisition. Real world videos exposed challenges and limitations to the techniques used at each stage of the system. These challenges were thoroughly explored in order to give a better insight of the noise sources that modern computer vision systems have to overcome.

Key contributions of this paper are piecing together the entire computer vision system, and getting it to produce meaningful results in challenging real world scenarios. Delving deeper into each of the three stages, unique contributions are made along the way. In the target acquisition stage, the novel DABS technique proved better then traditional ABS by 20% RMS Error for the long line video. In target tracking, the two layers of identifiers promote stability, and subdue the problem of identifiers being born too frequently because of poor Kalman filter predictions. Finally a method for creating a dynamic line model is proposed that is capable of dynamically determining parts of the image where a line forms.

REFERENCES

[1] Sharma, V., "A blob representation for tracking robust to merging and fragmentation," in *Applications of Computer Vision (WACV), 2012 IEEE Workshop on* , vol., no., pp. 161-168, 9-11 Jan. 2012

[2] Hare, S.; Saffari, A.; Torr, P.H.S., "Struck: Structured output tracking with kernels," in *Computer Vision (ICCV), 2011 IEEE International Conference on* , vol., no., pp. 263-270, 6-13 Nov. 2011

[3] Zivkovic, Z., and Ferdinand van der Heijden. “Efficient adaptive density estimation per image pixel for the task of background subtraction”. Pattern recognition letters, 27(7): 773–780, 2006.

[4] Zivkovic, Z. “Improved adaptive gaussian mixture model for background subtraction”. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, volume 2, pages 28–31. IEEE, 2004.

[5] Dalal, N., and B. Triggs. "Histograms of Oriented Gradients for Human Detection." 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)

[6] Xiu, Chunbo, Shian Wei, Rongfeng Wan, Yi Cheng, Jing Luo, and Huixin Tian. "CamShift Tracking Method Based on Target Decomposition." *Mathematical Problems in Engineering* 2015

[7] Comaniciu, D., and P. Meer. "Mean Shift: A Robust Approach toward Feature Space Analysis." IEEE Transactions on Pattern Analysis and Machine Intelligence IEEE Trans. Pattern Anal. Machine Intell. 24.5 (2002)

[8] Bradski, G.r. "Real Time Face and Object Tracking as a Component of a Perceptual User Interface." *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV'98 (Cat. No.98EX201)*

[9] Xu, Gang, Dong Zhao, Qi Zhou, and Ding Huang. "Moving Target Tracking Based on Adaptive Background Subtraction and Improved Camshift Algorithm." *2012 International Conference on Audio, Language and Image Processing* (2012)

[10] Exner, David, Erich Bruns, Daniel Kurz, Anselm Grundhofer, and Oliver Bimber. "Fast and Robust CAMShift Tracking." *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops* (2010)

[11] Li, Changyan, Lijun Guo, and Yichen Hu. "A New Method Combining HOG and Kalman Filter for Video-based Human Detection and Tracking." *2010 3rd International Congress on Image and Signal Processing* (2010)

APPENDICES

Appendix A: Code and User Guide

The code for the project can be found in the following github repo:

https://github.com/LineMonitoringProject/cv_algorithm

If you wish to be added to the project, email george.gargov@gmail.com

The code is broken up into several folders, each representing the three distinct stages of the algorithm. The segmentation folder contains the target acquisition code. In the segment.cpp file, ABS and DABS is used, along with connected component analysis. Then in the identify folder, all the material for target tracking is present. There are classes for Blobs, Identifiers, and Scores. The identify.cpp file contains the three stage matching process. The LineLogic folder contains all the code for dynamically determining the line region. Each of these stages is created into it's own thread, in main.cpp. Additionally main.cpp holds all the logic for the command line interface used by the user to launch the program.