

Robust Video Tracking Through Multiple Occlusions

Can a computer program be created to track a moving object in a video sequence, and successfully predict its locations as it travels through occlusions?

Copyright © 2009 - TJ Wilkason

Mount de Sales Academy

11th Grade

Warner Robins, GA 31088

Table of Contents

Abstract.....	1
Introduction.....	2
Purpose and Goals	3
Project Description	4
Applications.....	4
Materials and Methods.....	5
Materials.....	5
Research and Design Methodology	5
Testing Methodology	7
Overall Process Flow	9
Template Matching	12
Color Spaces.....	13
Conversion from RGB to HSV	13
Texture	14
Back Projection and Target Histogram.....	17
Background Ratio Histogram	18
Center Weighted Target Histogram.....	18
Dynamic Histogram Updating.....	19
Continuously Adaptive Mean Shift Tracking (CAMShift)	21
Mean Shift Theory	23
Mean Shift and CAMShift Algorithm.....	24
Kalman Filtering.....	26
Background - Process to be estimated.....	26
Kalman Filter Equation	27
Discrete Kalman Filter Equation	28
Results and Discussion	30
Kalman Filter Tuning	30
Kalman Accuracy and Kalman Convergence Cycle Reduction.....	31
Occlusion Detection	33
Improvements due to Saturation and Local Binary Patterns.....	38
Conclusion.....	42
Continued Research	43
References/Literature Cited	44
Acknowledgements	46
Appendix A - Software Listing	47

Table of Figures

Figure 1: Overall Process Flow.....	12
Figure 2: Various Templates for Early Target Detection.....	13
Figure 3: HSV and HSB Color Spaces.....	13
Figure 4: LBP Example.....	15
Figure 5: LBP Example.....	16
Figure 6: LBP Image Representation.....	17
Figure 7: Back Projection Example.....	17
Figure 8: Epanechnikov Center Weighting Function.....	19
Figure 9: Back Projection processing.....	21
Figure 10: Probability Distribution related to Pixel Values	22
Figure 11: Overall Description of CAMShift	25
Figure 12: Kalman Predict/Correct Cycle.....	29
Figure 13 & 14: CAMShift Convergence Iterations Mean and Kalman Tracker Accuracy.....	31
Figure 15: CAMShift Window Placement Accuracy for Kalman and non-Kalman case.....	32
Figure 16: Cumulative Convergence Cycles for Kalman and non-Kalman case.....	33
Figure 17: Occlusion Detection Logic based on CAMShift Area Rate of Change	34
Figure 18: Slow Kalman Filter Performance and Occlusion Detection	35
Figure 19: Occlusion Track Angles	36
Figure 20: Occlusion Frame Sequence – Wood Track Top View.....	37
Figure 21: Occlusion Frame Sequence – Cement Track.....	37
Figure 22: Occlusion Frame Sequence – The Author Walking behind and Occlusion	38
Figure 23: Improvements Due to Saturation + LBP.....	40
Figure 24: Back Projection – Improvements Due to Saturation + LBP	41

Abstract

Target tracking is increasingly becoming an important topic in the fields of surveillance, factory automation, and information gathering, as it identifies and tracks a certain object based on its characteristics. The purpose of this project is to expand the field of target tracking by creating a computer vision program that acquires and tracks objects in a real-time video sequence, both with and without occlusions. The goal of this project is to create a program that is able to predict the location of the object through the occlusion and continue tracking when the object exits the occlusion. Template Matching, Histogram Back Projection, and CAMShift algorithms are used to locate an object within a video sequence, and multiple Kalman filters are used to track and predict the location of the object as it moves around a scene that contains occlusions. The program was tested by using a number of video sequences of a robot maneuvering around a track, both with and without occlusions. The algorithm was designed to track the objects under various conditions including recorded and live video. The program measurements include the number of CAMShift convergence cycles with and without the Kalman filter, the error of the Kalman filter, evaluating the occlusion tracking performance, and finding the optimum process and measurement noise values. The results show the tracker is most efficient with the Kalman filter, which typically has an error of one pixel. With the filter, the project efficiently tracks the object through multiple occlusions and consistently reacquires the target at the exact position where it exits the occlusion. This algorithm expands the abilities of artificial tracking by creating an efficient method of tracking through occlusions. This project has a variety of applications, including surveillance and security, as well as any other field of study where locating and recording the location of objects is useful.

Introduction

Object tracking in video sequences has come about as a field of research recently due to the rapid growth of computer processing ability. In its simplest form, tracking can be defined as the problem of estimating the trajectory of an object in a video sequence as it moves in a scene. Tracking is performed by creating an algorithm that analyzes the individual frames of the video and finds the object in each frame, thereby continuously tracking it as it moves. Algorithms typically find the object based on specific features such as color, shape or size. However, when the object cannot be found in the frame, such as when it is blocked or occluded, many algorithms fail and the object can no longer be tracked. This is an issue that has created a challenge for the field of image analysis and object tracking.

There have been a variety of methods of object tracking [8] [19] and many attempts to overcome the problem of the occlusions. Some of the most popular methods of tracking include blob tracking [5], which is tracking based on a “blob” of movement based on a certain shape, such as a person, mean-shift tracking, which calculates the center of mass based on object criteria and shifts the tracker whenever the center of mass changes, and Lucas-Kanade tracking, which is a method of optical flow that measures the visual motion between two frames.

This project uses CAMShift tracking [7] [4], which is an improvement upon mean-shift [12] by allowing variation of the size of the track window. Of these various trackers [2], CAMShift was chosen for this project because its implementation is efficient, and it allows the tracker to continuously measure the change in motion of a specific object, while still allowing variability in the size of the window due to the shrinking of the object when entering an occlusion.

Initial CAMShift trackers [7] employed a single channel histogram comprised of the hue, or color, of the object to be tracked. To improve the robustness of the single channel, additional channels [4] were employed along with hue, including saturation and value, as well as Weighted Histograms and Background Ratio Histograms that account for the background colors. To account for dynamic changes of the target characteristics in the scene, the concept of target and candidate models [9] [10] were introduced to allow for dynamic updating of the target model over time [16].

This project employs the techniques above as well as the innovative use of texture for one of the dimensions of the Histogram, in order to improve tracking of objects with similar colors as their background. Various techniques to model texture [3] have been tried; however, they commonly require great computing resources and do not operate in real time. The technique used in this project to denote texture is called the Local Binary Pattern [13] [14], which has proven to be very

useful in the field of pattern recognition. However, there has been little prior research in the application of Local Binary Patterns in the field of target tracking.

To track the objects while they are occluded, “estimators” or “tracking filters” are needed. These estimators [15] are used to predict where objects will be in the future based on the filter parameters and prior locations of the object. Estimators also help to keep track of specific objects when nearby similar objects, and make sure the tracking is done as efficiently as possible. The two most common methods of combining tracking filters and video tracking are Kalman filters and Particle filters [1] [11] [17] [20] [21] . Kalman filters use the previously measured position of the object and calculate a normal probability function to identify where the object will likely be in the next frame. The condensation algorithm, one of the more popular Particle filters, works in a similar way, but is able to calculate multiple probability densities at once for multiple parameters, as opposed to the Kalman which can only calculate one. However, the Kalman filter [15] is more efficient and has been shown to perform equally well for this purpose, so it is therefore used for this project. Tracking estimators are critical components for tracking a target while it has become occluded, or while it is temporarily invisible to the tracking algorithm. The use of tracking estimators allows the algorithm to maintain a statistically valid estimate of the location of the object even while it cannot be “seen” by the tracker.

This project uses the combination of Kalman filters and CAMShift tracking along with an innovative occlusion detection and recovery algorithm to maintain track of objects throughout a video sequence.

Purpose and Goals

The purpose of this project is to create an efficient method of smoothly and continuously tracking an object in a video or real-time, including through occlusions and obstacles. The goal of this project is to design and implement a combination of the Back Projection, CAMShift, and Kalman estimator algorithms to track objects in a video sequence or in real time through multiple occlusions, and to reacquire the objects when they reappear. The project intends to overcome the loss of tracking ability due to occlusions and improve upon past methods of object tracking by using the Kalman filter to predict where the object will be after an occlusion, as well as by using texture to separate the target from a similarly colored background.

Project Description

This project involves creating a program that is able to efficiently track an object as it moves, even if it is blocked. The tracking is done by the CAMShift algorithm, which works by measuring the overall particle density of the object, and then once the center of density shifts, the function creates a vector that shifts the focus of the tracker to the new center of density. This function keeps the tracker continuously focused on the object as it travels around the track.

Kalman filters are used to keep track of the object location and predict the location of the object in subsequent frames to help the CAMShift algorithm locate the target in the next. Another Kalman filter is used similarly to track the target as it passes under occlusions by using the velocity and position of the object as it becomes occluded to maintain a search region for the CAMShift function as the target reappears from the occlusion. A third Kalman filter is used to track the area returned by the CAMShift algorithm and monitor the change in area to detect occlusions early.

Applications

This topic is important because it has a variety of applications, including security, for surveillance and recognition and tracking of criminals, military, for reconnaissance and other forms of visual information gathering, computer vision in factories, autonomous robots, computer interaction for gesture recognition, and traffic monitoring. This project would be applicable to any field that would be improved with advanced target tracking that could locate and record the location of objects. This project improves upon the field of computer vision by creating a method of automated visual tracking and prediction of occluded maneuvering objects. This particular aspect of computer vision is relatively new with much academic research, but with few recent applications.

Materials and Methods

Materials

The software used for this project included the OpenCV [6] version 1.1 software source code and Visual C++ 8.0 Express Edition in order to build the software. Microsoft Excel was used for the data collection and analysis. Hardware includes a webcam to track live video and a Digital Video (DV) video camera to capture test video. A Pololu 3pi robot programmed to follow a predefined path was used as a test target for the analysis portion of this project.

Research and Design Methodology

This project began by researching many computer vision papers on methods of object tracking. Questions were raised such as which object representation is suitable for tracking? Which image features should be used, including color and texture? How should the motion, appearance and shape of the object be modeled? How will the object be tracked when it is occluded? How will the object be reacquired once it exits the occlusion?

Object shape considerations include object points, such as corners or centroids, primitive geometric shapes such as circles, triangles, or rectangles, and object silhouette and contour, such as the boundary of an arbitrary object. More complex models include articulated shape or skeletal models where portions of the object move independently [19] .

Research into the various ways to represent the appearance of the objects included probability density estimates, object templates, active appearance models, and neural network object classifiers.

Based on the research and the desired outcome, the following criteria were created: 1) Target would be of a consistent color and shape; 2) Target would be autonomous and maneuver around a track; 3) Target would have multiple perspectives; 4) Target would be occluded much of the time and therefore invisible; 5) must allow for the background to be changed (non-fixed camera); and 6) the process must be able to operate in real-time on a laptop computer.

Based on the research and criteria, the Continuously Adaptive Mean Shift (CAMShift) algorithm was selected as the primary method to track the test target. The CAMShift algorithm, as used here, uses an image built upon the histogram of the object characteristics including hue, saturation, and texture, independent of target shape or aspect, as the primary criteria. The CAMShift algorithm is relatively new, but is well established as a viable tracker. Moreover, it has a relatively

efficient implementation which can operate in real-time. Since the goal was to track targets through occlusions, a method to estimate the target position while the target was occluded was required. The Kalman filter was chosen for its robust capability and efficient operation compared to other estimators.

For the software toolset, the OpenCV software library was chosen for its variety of functions and capability for image and video analysis. The OpenCV library was downloaded onto the computer, and a number of test programs were created and compiled using Visual C++ 8.0 and tested with the webcam to become comfortable using the complex software toolset.

Once the initial code was set up based on the initial idea of the project, a video was recorded of a robot with a blue arrow on top of it moving around a track a few times. This video was used to set up the specific parameters of the code that would be necessary to track the object. Once the video read into the computer, the initial object tracker was created in order to focus on the robot as it moved around the track. The tracker was created by using a number of OpenCV and custom algorithms include color channel conversions, histogram creation, back projection, CAMShift function, and a template matcher that was used to initially locate the target in the first frame.

Once the tracker was created, occlusions were added in order to determine how the Kalman filter would be needed to predict the location of the object once it could no longer be seen. On an initial trial, it was found that as the object traveled under the occlusion, the track box became smaller and the measured velocity slowed down, as the center of mass was constantly slowing down as the target shrunk under the occlusion. The Kalman predictor continued to slow down, and thus was not able to predict accurately where the object would be once it left the occlusion. This was corrected by creating a method to detect the occlusion early, so the Kalman filter would be able to move at an accurate speed and in an accurate direction to track the object efficiently. Occlusion handling was added with the combination of an occlusion detector and occlusion tracker. The occlusion detector uses a Kalman filter on the size of the target, which detects a rapid decrease in the target area, which is associated with an occlusion. An occlusion Kalman filter, which runs slower than the fast Kalman filter, keeps track of the target through the occlusion, allowing the CAMShift to reacquire the target as it becomes visible.


The code was further modified to handle background noise that would often interfere with the CAMShift centering the target properly. This was partially solved by automatically adjusting the Saturation and Luminance histogram thresholds by limiting them to within one standard deviation of the target Saturation and Luminance. It was later improved by using a background weighted histogram of the object rather than a simple histogram of the object.

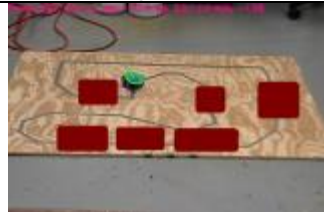




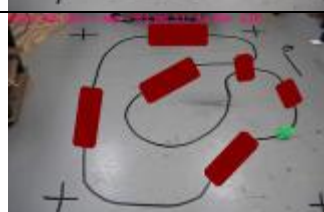


Once the code was optimized to track objects through occlusions, tests were run and data was logged to determine specific parameters of the code and the efficiency of the code itself. Artificial occlusions were added to the video sequences to test the performance of the code. The code was modified to log the values of the variables for each frame to allow offline quantitative performance evaluation. The cumulative number of convergence cycles for the tracker with and without the Kalman filter was measured, along with the values of each of the Kalman filters and the value of the detection of the occlusion, to measure the efficiency of the detection method. Also, to find the optimum use of the process and measurement noise, the values of the convergence cycles and the error of the Kalman filters for each video with varying process noises from 1 to 15 and measurement noises from 0 to 8 were measured and put in a 3D graph to find which values were optimum.


Following data collection on test objects, the tracker was used on a number of real-life video sequences and was further modified to add the saturation channel and a method to discriminate texture called Local Binary Patterns. With the change from a 1D histogram and single hue channel to a 3D histogram with hue, saturation and LBP [13] [14] index, the performance of the tracker improved greatly for complex situations where the object is near other similarly colored objects.

Testing Methodology

Testing of the Kalman enhanced CAMShift program was performed with the following video capture sequences described in the table below. These videos were all recorded by the author, using a Pololu 3pi robot as the target. The Pololu robot was programmed via the Arduino environment as a "line follower." This algorithm uses sensors in front of the robot to track adhesive tape placed on the surface. The robot tracks the tape as it moves along the course. The robot video was captured with a Panasonic DV camera, and the DV video was downloaded and processed on the computer using the open source MEncoder program to put it into a format readable by the OpenCV library. This test method proved itself very useful in establishing repeatable tests. Other videos were recorded for testing, including one of the author moving behind a board and multiple videos of skaters around a short track.

Video Sequence	Description
	Side view on Wooden Track, initial video used to test and debug the basic tracking portion of the program. Used for testing of the basic Kalman tracking performance.

Video Sequence	Description
	Side view on Wooden Track, initial video used to test and debug the basic tracking portion of the program. Simulated occlusions added to test occlusion performance.
	Top view on Wooden Track, same track as above but with a different view. Also used to test and debug the program. Used for testing of the basic Kalman tracking performance.
	Top view, on Wooden Track. Same video as above, however six 90 pixel square occlusions are placed around the track. These occlusions are overlaid on the image prior to processing so they operate exactly as a real occlusion would. The bottom four occlusions are used to test the extreme performance of the Kalman tracker.
	Top view, on Cement Floor Track. Used for testing of the basic Kalman tracking performance under more extreme conditions. The track is larger and thus the target is smaller. The target becomes very small at the top of the track.
	Top view, on Cement Floor Track, with real occlusions. Tunnels were added to the track to validate the performance of the tracker under real lifelike conditions. The small target and relatively large occlusion size would not be too different from a car on roadway or small object on a factory floor.
	Top view, on Cement Floor Track, with artificial occlusions. This is the most challenging video. 90 Pixel artificial occlusions were placed on this image and in very difficult locations, including corners and the very top of the image. This is the most challenging test for the tracker under laboratory conditions.
 <small>Frame: 135, Conv: 0, Ang: +178.0 Vel: 3.7 ACQ: +01.4%</small>	Author walking in circles around a large occlusion. Used to judge the program using real life pedestrian sequences.
 <small>Frame: 66, Conv: 2, Ang: +108.7 Vel: 1.0 ACQ: +13.0% BH: 0.71 SHORTTRACKHD.COM</small>	Short Track Video #1, skater is followed by a camera as they skate around a track for a number of loops. Mild occlusions with the skater being occluded by other skaters and fans standing up. Permission to use video sequences from Tony Chung at SHORTTRACKHD.COM .

Video Sequence	Description
	<p>Short Track Video #2. This is the acid test for the program. Tracks a skater at a much lower camera angle with multiple occlusions as the skater changes position with other skaters, and the skater passing behind judges quite often. To further complicate the sequence, there are a number of similar colored objects in the video both in the stands and a large red banner as shown to the left. This was the primary video used to develop the addition of the LBP and saturation channels. Permission to use video sequences from Tony Chung at SHORTTRACKHD.COM</p>

Formal testing consisted of CAMShift tracking accuracy, both with and without the Kalman filter engaged, tuning of the Kalman filter parameters, and recording and tracking performance through occlusions. The testing was performed using the video sequences above. The test program used a template of the test target to allow the program to identify and lock onto the target in the first video frame. This automation allowed repetitive testing in a controlled environment as well as easing the tuning and troubleshooting of the program. Additional testing was performed using a Webcam and live targets to prove the program worked with other objects. However, using live video makes it difficult to set up and repeat specific tests.

Overall Process Flow

Figure 1 on page 12 shows the overall process. It begins by initializing the parameters of the program, the template images, all the variables and functions, the selection of the region of interest if the template doesn't match, the capture and initialization of the video from the input, and the creation the Kalman filters. The loop then begins by reading the frame, either from a specified video file or from a camera, and attempting to locate the target from each of the template images. If the template best match is greater than 90%, then an initial region selection is automatically performed at the target location. Otherwise, the next frame is read from the camera or video file. The template match makes an attempt in the first two frames to find a match.

The program then checks to see if a selection region has been performed, either automatically from the template match or from a user selection. If not, the program continues to read frames, waiting for a later selection. Once a region has been selected, the tracking flag is set to Initialized. The selection region, selected by either the user or the template matcher, is called the Region of Interest (ROI), and consists of a set of rectangular coordinates around the object.

If tracking mode is enabled, the saturation and value channels are restricted to certain ranges, with the minimum saturation and value (luminance or brightness) channel thresholds set to 1 standard deviation above and below the average values for the selected object. This threshold is used

to limit “noise” in the image due to gray or unsaturated colors. The hue, saturation and value channels are then saved as individual channels to process. The hue channel is used because it contains the colors of the target, independent of saturation (purity) and luminance (brightness). The saturation channel shows the purity of the color to help distinguish between different saturation values of the same hue. The value channel represents the gray scale portion of the image, showing only brightness. This value channel is used later by the Local Binary Pattern algorithm.

If Initial Track mode is set, the hue, saturation and value channels are masked by the Region of Interest (ROI), which discards all data except for the area under the target. This ROI is used to create a histogram later used by the back projection. Finally, the Kalman filters are initialized to the selection region.

If the tracking mode is on, the next step runs the value channel through the LBP algorithm, which replaces each of the pixels with the corresponding LBP index from 0 to 35. These indexes represent the texture found at that pixel location. Then, a three dimensional histogram comprised of the hue, saturation and LBP channels is created within the ROI. A corresponding three dimensional histogram of the background area outside the ROI (3X the ROI area) is created as well, called the background histogram. The foreground histogram values are weighted by the background values in such a way as to enhance the unique characteristics in the foreground and diminish the characteristics common to the foreground and background. This technique is called a Background Weighted Histogram, or Ratio Histogram. The resulting target histogram is scaled such that the most prominent bins have a value of 255. The target and background histograms have dimensions of 32 (hue) X 16 (saturation) X 36 (LB), for a total of 18,432 bins.

A back projection image is formed using the hue, saturation and LBP channels along with the 3D target histogram. The back projection image is a single channel image that contains pixels with probability factors from 0 to 255 that correspond to the probability of that pixel belonging to the target within the ROI.

The next step masks the channels with saturation or brightness values outside the standard deviation limits set during target histogram creation. This processed image is used as an input to the CAMShift function and should only have the target shown as white (or light grey) pixels and other regions show as black pixels. The Back Projection image pixels will have a value from 0 to 255 relative to the probability of the pixel within the three channels, as transposed by the histogram of the selected ROI. Figure 9: Back Projection processing on page 21 shows an example of this process, including a before and after image of the back projected image along with the hue, saturation and LBP images.

The CAMShift function is then called with the processed back projection image, along with a specific region of interest window. The CAMShift function operates on the image within the region of interest. The result of the CAMShift function is a set of coordinates with the target size, position, area, and rotation angle.

The occlusion detection logic then determines if the target is close to or fully occluded by checking the CAMShift target area and area rate of change. If the area is less than the threshold (50% of last known tracked value) or is decreasing at a rate of greater than 25%/frame, then an occlusion is detected.

If an occlusion is detected, the Kalman filters (target fast position, slow position and area) are updated with their own prediction. The search box for the next CAMShift operation is then increased in size based on the rate of change of the slow Kalman tracker, and the next image is read.

If there is no occlusion, the Kalman filter is updated with the CAMShift coordinates and then a prediction is made for the next frame. The prediction is used to center the search window for the CAMShift operation. A new candidate histogram is created for the current ROI and compared to the current target histogram using a Bhattacharyya Coefficient [10] . If the histograms are close, the target histogram is updated with information from the candidate histogram to accommodate changes in lighting.

This entire process is repeated until there are no more frames to process.

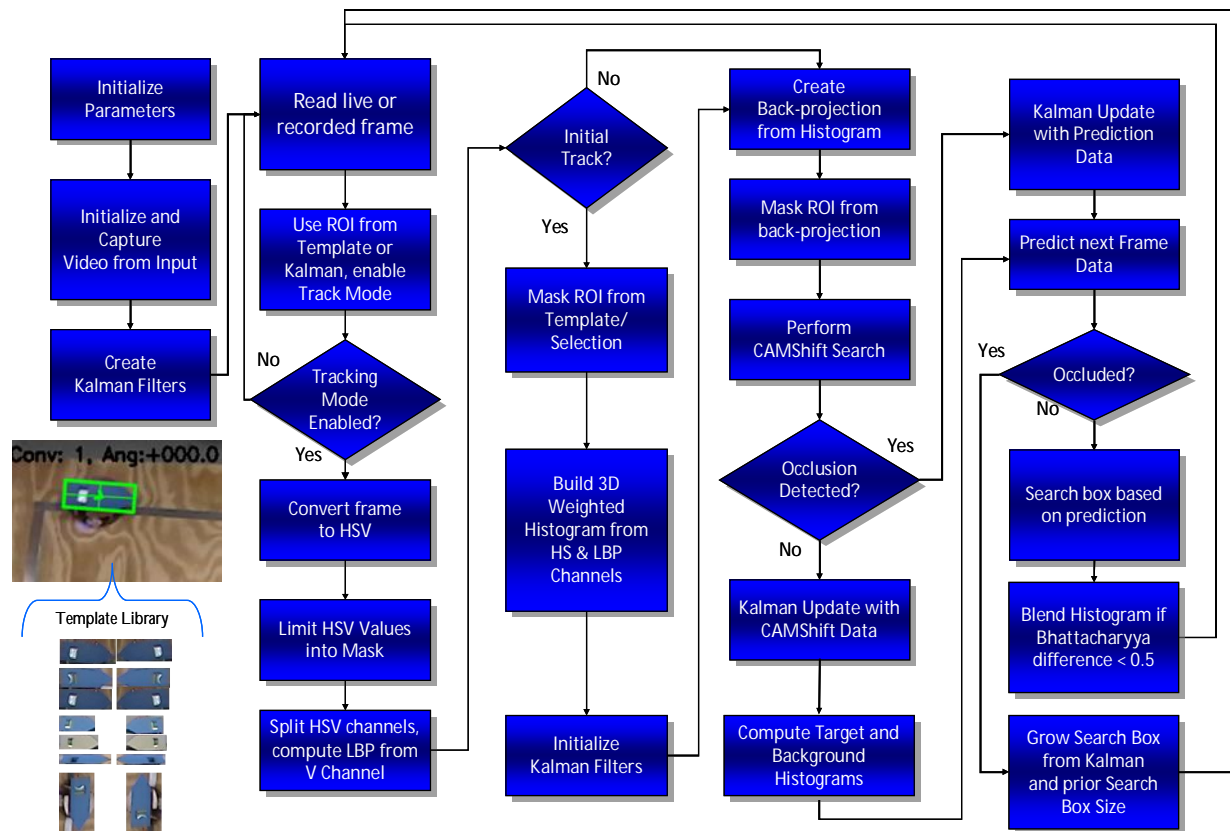


Figure 1: Overall Process Flow

Template Matching

Template matching is used in this project to automatically locate the object to be tracked without having to select it first. The program searches the templates folder for any images and attempts to match them against the first few frames of the video. Some examples of template images are shown in Figure 2: Various Templates for Early Target Detection below. The template matching process finds a maximum correlation value for each pixel in the frame for each of the images. When the image with the greatest correlation value is greater than 90% at a group of pixels, it places the region of interest in that area. The correlation values are found by adding up the similarities of each pixel as it shifts across. It does this for each pixel in the frame until it finds the region with the highest correlation value, with the pixels that have the highest similarity. Using the template library makes it easy to capture a region from the first frame, crop it, and save it in the template folder. This allows repeated testing of the algorithm.

Below are some of the templates used for this project.

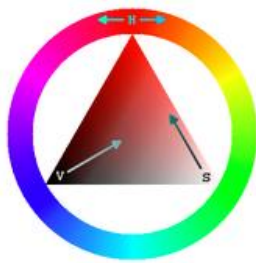




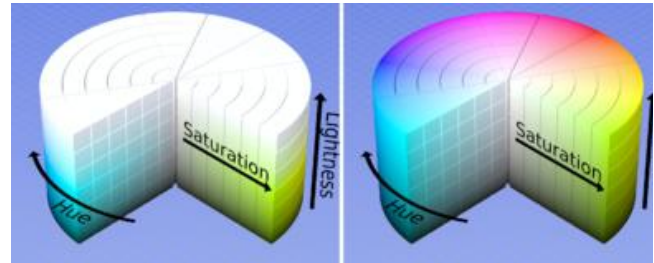
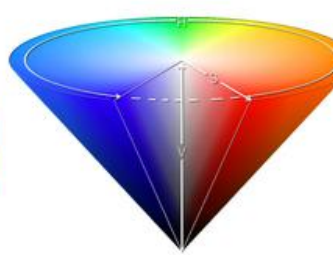
Figure 2: Various Templates for Early Target Detection

Color Spaces

Various color spaces are used outside of the regular red-green-blue to expand the various methods of color identification. They are created by using values for color, pigment, or properties of colors, and assigning these values to coordinate axes to create a shape or region of space that contains all possible combinations of the various values. Apart from RGB, there is the HSV color space, along with the similar HSB (Hue Saturation Brightness), and the YCbCr, which uses the luma, or brightness, and the chrominance for the red-difference and the blue-difference. In this project, the HSV color space is used to accurately identify the object as it moves around the track. The object is identified by its hue (i.e. color regardless of intensity or grayness), and saturation (i.e. pureness of the color). The hue and saturation values are split from the image and independently processed to form part of the histogram used for tracking. Figure 3 below shows some diagrams that represent the HSV and HSB color spaces.



HSV represented as a triangle and a cone. The hue is the ring around that shows all the different colors, the value is moving towards the center or the bottom, moving from light to dark, and the saturation is moving away from white to the pure color.



HSB and HSV represented as cylinders. The hue value is found by moving around the cylinder, the value, or brightness, is moving up and down through the cylinder, and the saturation is moving towards the center and out to the edge.

Figure 3: HSV and HSB Color Spaces

Conversion from RGB to HSV

Since the capture from the camera used the RGB color space, a conversion of each frame to HSV color space is needed. The algorithm used for the color conversion is as follows.

Let $r, g, b \in [0,1]$ be the red, green, and blue coordinates, respectively, of a color in RGB space. Let max be the greatest of r, g , and b , and min the least value. To find the hue angle $h \in [0, 360]$ for either HSL or HSV space, compute:

$$h = \begin{cases} 0, & \text{if } \max = \min \\ (60^\circ \times \frac{g-b}{\max - \min} + 360^\circ) \bmod 360^\circ, & \text{if } \max = r \\ 60^\circ \times \frac{b-r}{\max - \min} + 120^\circ, & \text{if } \max = g \\ 60^\circ \times \frac{r-g}{\max - \min} + 240^\circ, & \text{if } \max = b \end{cases}$$

The value of h is normalized to lie between 0 and 180° to fit into an 8 bit grayscale image (0-255), and $h = 0$ is used when $\max = \min$ (that is, for grays), though the hue has no geometric meaning for gray, where the saturation $s = 0$. Also, the choice of 0 as the value for s when l is equal to 0 or 1 is arbitrary.

The values for s and v of an HSV color are defined as follows:

$$s = \begin{cases} 0, & \text{if } \max = 0 \\ \frac{\max - \min}{\max} = 1 - \frac{\min}{\max}, & \text{otherwise} \end{cases}$$

$$v = \max$$

The v , or value channel represents the gray scale portion of the image and is used in the next section for Local Binary Patterns.

Texture

Although color is helpful in identifying and isolating objects, other characteristics of the object are needed to discriminate between the object and other similar colored objects. A useful discriminator is the texture of the image. The texture indicates the uniformity of the surface of the object, such as a flat surface, woven fabric, wood grain or other color independent qualities.

In the field of image recognition, a popular method has come about in recent years that is both efficient and effective. The method is called Local Binary Pattern, or LBP. LBP is a method by which the cells surrounding the pixel of interest (center pixel) are compared to the center pixel and based on the difference in magnitude between them, a binary pattern is formed with a 1 or 0 corresponding to each surround pixel that identifies if the pixel is greater than the center pixel (1) or less than or equal to the center pixel (0). The most efficient LBP pattern uses the eight surrounding pixels to a center pixel and creates a single byte value (8 bits) that represent the comparative values of the surround pixels to the center. For example, Figure 4 below shows a center pixel located at g_0 , with the eight surrounding pixels g_1 to g_8 .

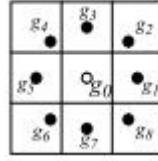


Figure 4: LBP Example

The equation for the resulting LBP pixel, denoted as LBP_8 for the 8 bits is thus:

$$LBP_8 = \sum_{i=1}^8 s(g_i - g_0) 2^{i-1}$$

Where the function $s(x)$ returns a binary value of 0 or 1.

The method described above works better on some images than others. One deficiency is that the local contrast of the image is not considered in the method; the method only measures if pixel values are greater than other pixel values. To account for relative image contrast, a method has been created to establish a threshold in the equation based on the image standard deviation. This accounts for the global contrast of the image. Calling this value the S_i , the equation can be rewritten as:

$$LBP_8 = \sum_{i=1}^8 s(g_i - (g_0 + S_i)) 2^{i-1}$$

The image used for the LBP is the value channel, which contains the grey scale version of the image, and replaces it with an image based upon the LBP equivalent and uses that for the Back Projection Histogram. For this project, the value of S_i was computed by taking the value plane standard deviation, in pixels, and dividing by a constant, $C=12$ in this case, which was found to be most effective after various trials. The constant value is arbitrary, but using 12 worked well for the images in this project.

$$S_i = \frac{StdDev(V(x, y))}{C}, \text{ where } C = 12$$

Using this constant as an offset, the algorithm will effectively put the same percentage of LBP values into the 0 value independent of the image contrast. The LBP value of 0, which is often the majority of the pixels, represents a blank solid texture.

This simple but effective method works very well for identifying textures. However, this method creates 256 possible different patterns. These patterns are sensitive to rotation; for instance, if a pattern of 0111 1111 is generated, and the image is rotated by 45°, the pattern would become either 1111 1110 or 1011 1111, completely different numbers but the identical texture.

To overcome this limitation, the idea of a rotationally invariant Local Binary Pattern [14] was researched. It turns out that if you take every number from 0 to 255, in binary, and rotate the bits (ROR) around, and save the highest value of the eight rotations, you will wind up with only 36 unique patterns, reduced from 256. This creates what is called a rotational invariant local binary pattern, or LBP_8^{ri36} . This is mathematically identified as:

$$LBP_8^{ri36} = \min \{ROR(LBP_8, i) \mid i = 0, 1, \dots, 7\}$$

Visually, the resulting 36 patterns are those shown in

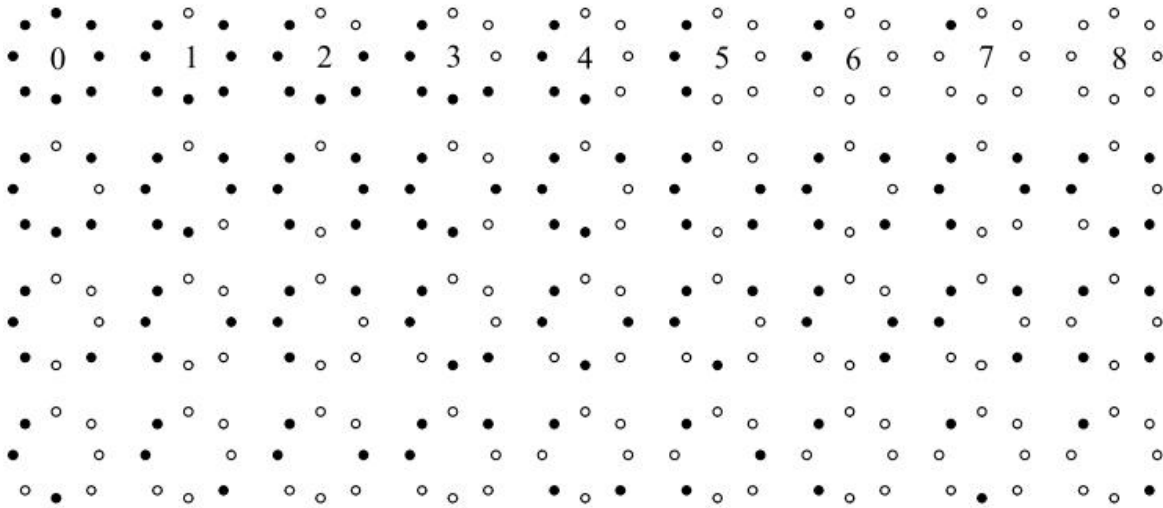


Figure 5: LBP Example

There are further reduced patterns you can create such as LBP_8^{ri9} , LBP_8^{ri5} that focus on the most structured patterns (lowest number of transitions). However, for this project the LBP_8^{ri36} method proved to be the most robust.

The method used to implement the LBP was to take the Value channel of the converted HSV image and run it through the LBP algorithm and replace the image plane with the LBP converted value. To visualize this, the following two images of Figure 6 are shown that show the V (or gray scale) plane and the converted LBP image. The LBP image will not only highlight those areas with the most contrast, but assign them specific codes based on the pattern of the contrast.



Figure 6: LBP Image Representation

Back Projection and Target Histogram

The back projection of the image is what helps the CAMShift identify the object to be tracked. The back projection algorithm computes a pixel value, on a single plane image, based on the combination of the passed in image planes (hue, saturation and LBP) and their corresponding bin counts in the histogram. This technique will create a single image plane, called the back projection image, where each pixel value corresponds to the probability that the pixel value in the original color image is actually a pixel in the target. To help explain the process, Figure 7: Back Projection Example shows the case where only two dimensions are used (rather than three).

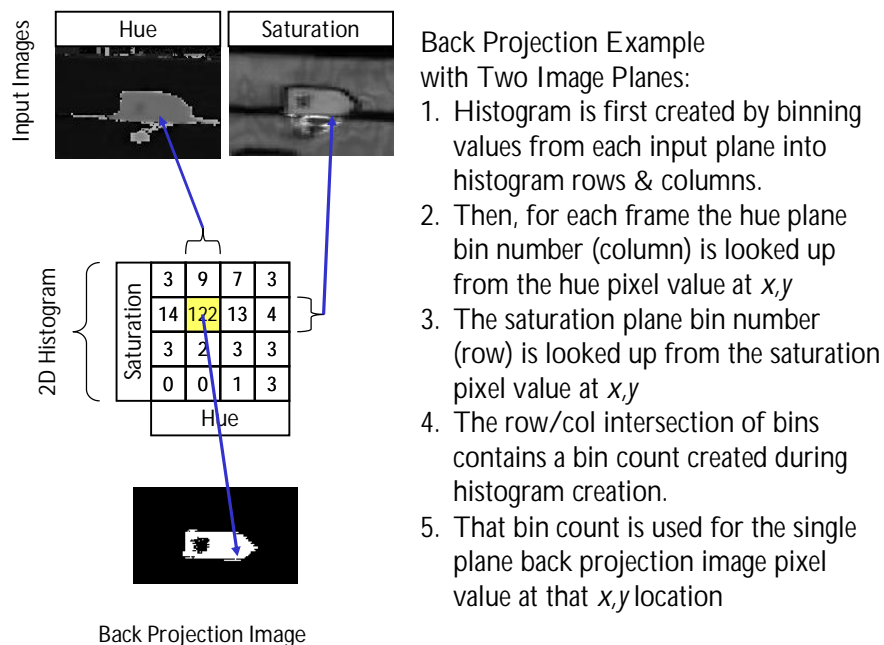


Figure 7: Back Projection Example

The target reference histogram is created during the selection process, where the selection region is identified automatically using an image template or by user selection, where the user selects the region with the mouse. The histogram process will categorize the value of each pixel in the selected region and put each into one of N bins, corresponding to N bins of the histogram dimension. In this case a three-dimensional histogram is used with dimension of 32 (hue) X 16 (saturation) x 36 (LBP) bins = 18,432 bins. In a similar way, a histogram is created for the remainder of the background to identify the predominant values not in the target. Weights are assigned for each of the target bins such that the target values unique to the target will have a higher relative value versus hues that are in common with the background. The resulting histogram lookup table maps the input image plane value to the bin count, and then normalizes from 0 to 255 to ensure the

resulting grey scale image has valid pixel intensity values. The initial target histogram lookup table is created at target selection and is saved as a reference histogram for later processing. The latest target histogram is then used for each frame to map the hue, saturation and LBP image planes into a resulting back projection image used by the CAMShift process. The resulting image from this process has mostly white pixels (255) at the predominant target locations and lower values for the remaining colors.

Background Ratio Histogram

Define the background bin weights for a total of m bins

$$b = b_n, n = 1 \dots m \text{ where } \sum_{n=1}^m b_n = 1 \text{ (normalized)}$$

From the background model a set of weights are defined to allow prominent features in the background to be diminished in the foreground histogram. A numerator from the background weights \hat{b} is selected as the smallest non-zero background histogram value. Then the feature weights as are computed as:

$$w_n = \min \left(\frac{\hat{b}}{b_n}, 1 \right)_{n=1 \dots m}$$

These weights are used to adjust the target model bin values to diminish background values and enhance values unique to the target. The new target model values f become:

$$f_n = f_n w_n$$

Center Weighted Target Histogram

To help ensure the target histogram likely contains more values from the target, rather than the surrounding background, the target histogram is created by accumulating bin values first on the entire selection region, then the window is made smaller and smaller in each dimension as more values are accumulated. This creates a center weighted histogram that provides for “weight” to the pixels near the center of the target (which are more likely those needed) vs. the pixels on the edge of the target (which are more likely those of the background). The window pattern used is based on the Epanechnikov formula widely used for Gaussian weighting. The formula is as follows

$$K(x) = (1 - x^2) \{ |u| \leq 1 \}$$

And has a shape like the following. As denoted by the shape, the pixels closer to the center will receive more weight than the pixels near the edge.

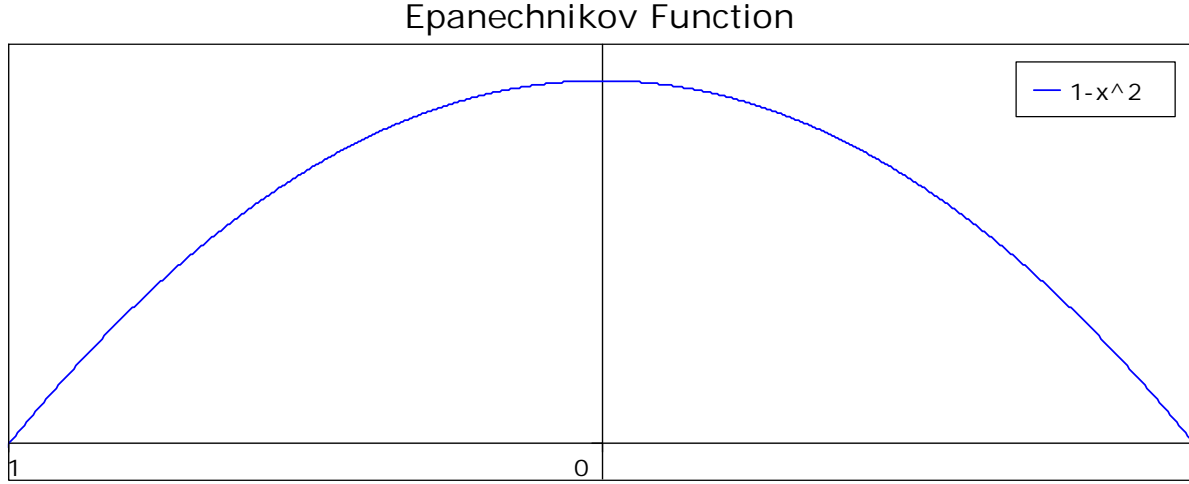


Figure 8: Epanechnikov Center Weighting Function

Dynamic Histogram Updating

After each frame is read and the logic determines there is no occlusion, the target histogram is updated to account for any illumination or color shift changes of the target. This is done by computing a candidate histogram of the target in the same manner the original target histogram is computed. Then the candidate and target histograms are compared using the histogram comparison technique that uses a Bhattacharyya Coefficient. There are a number of histogram comparison techniques, but the Bhattacharyya Coefficient is a commonly used technique in literature and works well for this purpose. The Bhattacharyya Coefficient will return a value from 0 to 1, with 0 being a perfect match and 1 being a total mismatch of the histograms. The equation for the Bhattacharyya Coefficient $d_{Bhattacharyya}$ that compares a target histogram H_t with a candidate histogram H_c is as follows:

$$d_{Bhattacharyya}(H_t, H_c) = \sqrt{1 - \sum_i \frac{\sqrt{H_t(i) \cdot H_c(i)}}{\sqrt{\sum_i H_t(i) \cdot H_c(i)}}}$$

If the Bhattacharyya Coefficient is less than a set number, 0.5 for this project, then the current target histogram is merged with the new candidate histogram to account for changes in lighting. The merging, or blending is per the following equation.

$$H_t = H_t \alpha_1 + H_c \alpha_2 + H_r \alpha_3$$

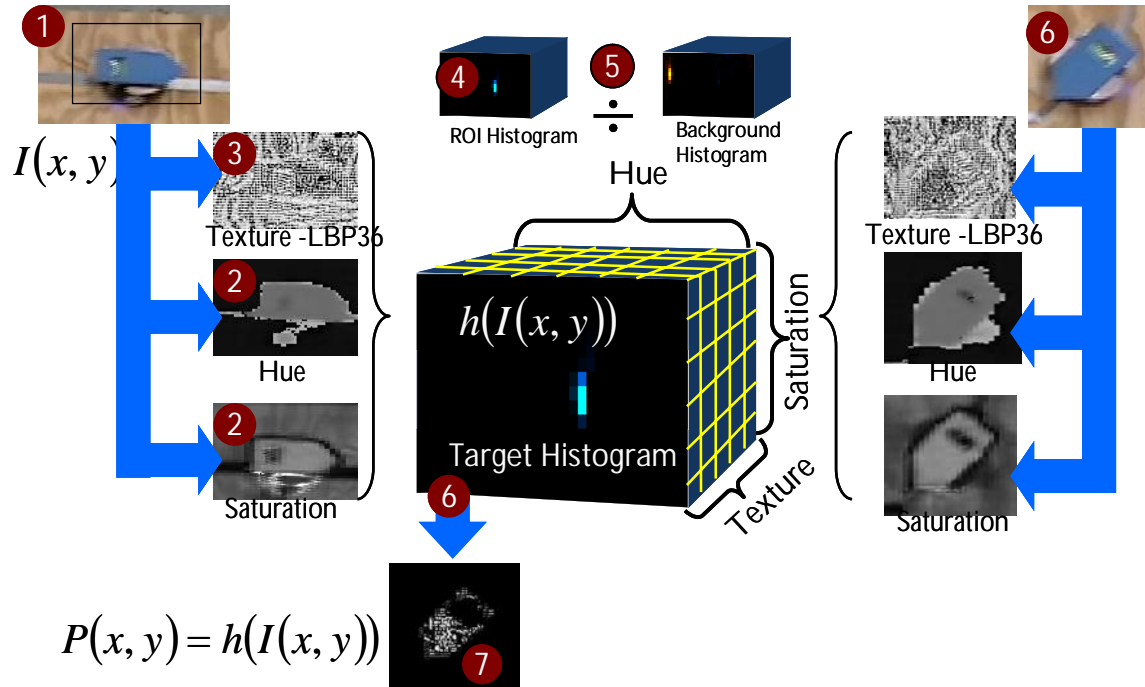
Where H_t is the current target histogram, H_c is the candidate target histogram just compute, and H_r is the original or initial target histogram, called the reference histogram. The blending terms are selected to allow the candidate histogram to shift the target histogram towards

the candidate histogram but ensure it does not shift too far off the original reference histogram. This blending is needed to ensure that spurious or temporary lighting changes do not cause the target histogram to drift too far away from where it should be. The blending values must ensure that $\alpha_1 + \alpha_2 + \alpha_3 = 1$ to maintain histogram integrity. The value B is used as a blending factor. For this project the following values were chosen:

$$B = 0.45, \alpha_1 = 0.95, \alpha_2 = (1 - \alpha_1)B = 0.0225, \alpha_3 = 1 - \alpha_1 - \alpha_2 = 0.0275$$

With these values, the target histogram will always maintain 95% of its original value and be slightly adjusted towards the candidate histogram and original target (reference) histogram.

Figure 9: Back Projection processing describes the processing steps necessary to map the Red Green Blue (RGB) image into a binary form suitable for the CAMShift operation.



Back Projection Processing

1. Region of Interest (ROI) selected by template or user.
2. Red-Green-Blue converted to Hue-Saturation-Value pixel planes.
3. Value Plane converted to Rotation Invariant Local Binary Pattern (LBP36) to capture texture.
4. 3D Histogram of hue, saturation & LBP (ROI Histogram) created mapping hue, saturation and LBP values into normalized bin counts. Background Histogram created with image data outside of ROI.
5. Ratio ROI and Background histogram into Target Histogram to enhance target unique bins. $h(I(x, y))$
6. During processing the image is again split into planes and the Back Projection maps image planes into probability map using target histogram to highlight pixels most likely in target. $P(x, y) = h(I(x, y))$
7. Resulting image has high value (white) areas for hues of interest, all other areas black. Probability image ready for CAMShift operation.

Figure 9: Back Projection processing

Continuously Adaptive Mean Shift Tracking (CAMShift)

The CAMShift algorithm, which stands for continuously adaptive mean-shift, is a method that is used to center a bounding region on an image that maximizes the correlation between the image and the probability distribution in the back projection histogram.

The inputs to the CAMShift algorithm are window coordinates to restrict the search, and a back projection image where the pixel values represent the relative probability that the pixel contains the hues, saturation and Local Binary Pattern (LBP) indexes of the object of interest.

The CAMShift is an extension of the Mean Shift algorithm. The difference is that at the end of the Mean Shift computation, the object size and aspect ratio are computed and returned to the program. The unique CAMShift capability allows a program to continuously account for the object size and aspect ratio as it is tracked. The CAMShift tracker works by iteratively calculating the overall center of mass of the object within the ROI region of the probability distribution, and it then returns the location, size and orientation of the mass. To find the center, the algorithm computes the moments M_{00} , M_{10} , M_{01} of the selected region. The center, or centroid, is computed by the ratio of the M_{01} and M_{10} moments to the M_{00} moment. These ratios return a gradient vector which denotes where to reposition the search window for the next iteration. The tracker has converged on the center of the object once the vector is equal to zero. The process is iterative as it continues to re-center the track window for as long as it takes to converge upon the object. It will always find the center. The CAMShift function improves upon the Mean-Shift by allowing the size of the returned window to fluctuate, allowing the object to be tracked more accurately as it moves away from or towards the camera, as the size of the window would need to be changed to keep total track of the object. Moreover, the CAMShift function returns the size, aspect, and orientation of the window to be used by the program.

Figure 10 below graphically shows the mapping of the back projection pixel values to the underlying probability density function. The algorithm will locate the highest center of mass peak in the search window.

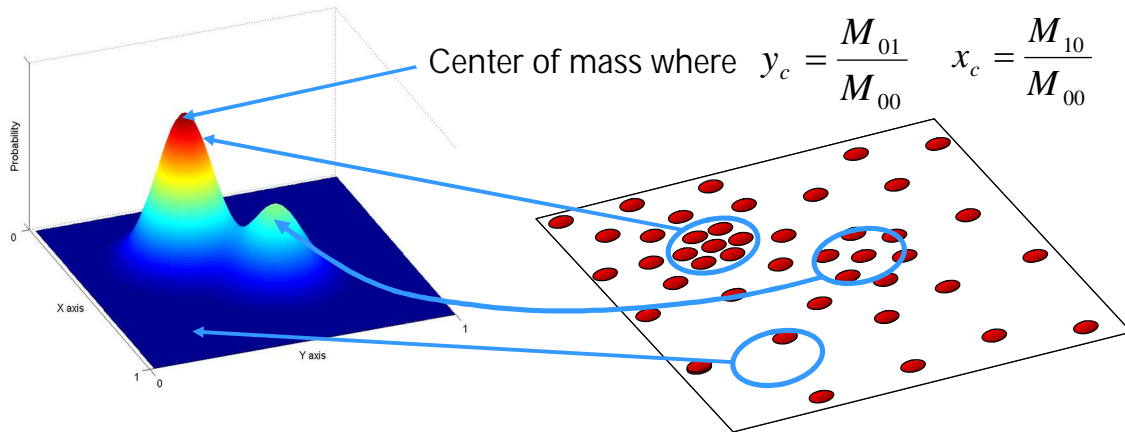


Figure 10: Probability Distribution related to Pixel Values

Mean Shift Theory

The original Mean Shift algorithm is based on the use of kernel density functions and weighted averages of the underlying data. Given n data points, $x_i = 1 \dots n$ on a d -dimensional space R^d , the multivariate kernel density estimate obtained with kernel $K(x)$ and window radius h is

$$f(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) \text{ with probability distribution } P(x) = \frac{1}{n} \sum_{i=1}^n K(x - x_i)$$

For radially symmetric kernels, such as a Gaussian kernel, the kernel function can be defined as

$$K(x) = ck\left(\left\|x\right\|^2\right)$$

Where c is a normalization constant which assures $K(x)$ integrates to 1. The mass center of the density function are located at the points where the gradient is zero, or $\nabla f(x) = 0$

The gradient of the density estimator is

$$\begin{aligned} \nabla f(x) &= \frac{2c}{nh^{d+2}} \sum (x_i - x) g\left(\left\|\frac{x - x_i}{h}\right\|^2\right) \\ &= \frac{2c}{nh^{d+2}} \left[\sum_{i=1}^n g\left(\left\|\frac{x - x_i}{h}\right\|^2\right) \right] \left[\frac{\sum_{i=1}^n x_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^n g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x \right] \end{aligned}$$

Where $g(s) = -k'(s)$. The first term is proportional to the density estimate at x computed with kernel $G(x) = cg\left(\left\|x\right\|^2\right)$, and the second term

$$m_h(x) = \frac{\sum_{i=1}^n x_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^n g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x$$

Is the mean shift, i.e. the vector difference between the weighted average of the window location (left hand term on right side of equation) and the prior center of the window x . The mean shift vector always points toward the direction of maximum increase in the density. The mean shift procedure always converges after some finite number of iterations such that the $m_h(x)$ function is zero.

Mean Shift and CAMShift Algorithm

For the case of a discrete 2D image probability distribution, the computation is greatly simplified mean location (x_c and y_c) can be found using the moments of the back projection image as shown below

$$x_c = \frac{\sum_x \sum_y xP(x, y)}{\sum_x \sum_y P(x, y)}; y_c = \frac{\sum_x \sum_y yP(x, y)}{\sum_x \sum_y P(x, y)}$$

Where $P(x, y) = h(I(x, y))$ is the back projected probability distribution at points x, y within the search window $I(x, y)$ that is computed from the histogram h of I . Or simplified to

$$x_c = \frac{M_{10}}{M_{00}}; y_c = \frac{M_{01}}{M_{00}}$$

where

$$M_{00} = \sum_x \sum_y P(x, y) - \text{Zero'th moment,}$$

$$M_{10} = \sum_x \sum_y xP(x, y); M_{01} = \sum_x \sum_y yP(x, y) - \text{first order moments}$$

Image moments are a weighted average of the image pixel intensities which have useful properties to return the centroid, size and aspect information of a probability distribution.

These positions and dimensions of the search windows are updated at every iteration of the process until they converge. Once convergence is complete, the first two Eigenvalues (major length and width) of the probability distribution found by CAMShift are calculated as follows

$$\text{Let } a = \frac{M_{20}}{M_{00}} - x_c^2, b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right) \text{ and } c = \frac{M_{02}}{M_{00}} - y_c^2$$

Where the second order moments are defined as

$$M_{20} = \sum_x \sum_y x^2 P(x, y); M_{02} = \sum_x \sum_y y^2 P(x, y)$$

This creates a covariance matrix of

$$\text{cov}(P(x, y)) = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

The length l and width w of the covariance matrix represent the Eigenvalues and can be computed as

$$\lambda_i = \frac{a+c}{2} \pm \frac{\sqrt{4b^2 + (a-c)^2}}{2}$$

or

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \quad \text{and} \quad w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}}$$

The object orientation (major axis) is

$$\theta = \frac{\arctan\left(\frac{b}{a-c}\right)}{2}$$

Figure 11: Overall Description of CAMShift below shows the relationship of the image processing and the math behind the CAMShift operation. The M_{00} , M_{10} and M_{01} moments are computed each cycle and the resulting gradient vector is used to re-center the search window for the next iteration. When the slope becomes zero (i.e. the vector length is zero) the window is considered centered and the iteration stops. Following the convergence of the window the M_{20} and M_{02} moments are computed which are used to compute the target size and aspect information.

For this project the initial search window is estimated using a Kalman filter which greatly reduces the number of iterations required to center the window. Figure 15 and Figure 16 starting on page 32 show the improvement due to the Kalman Filter

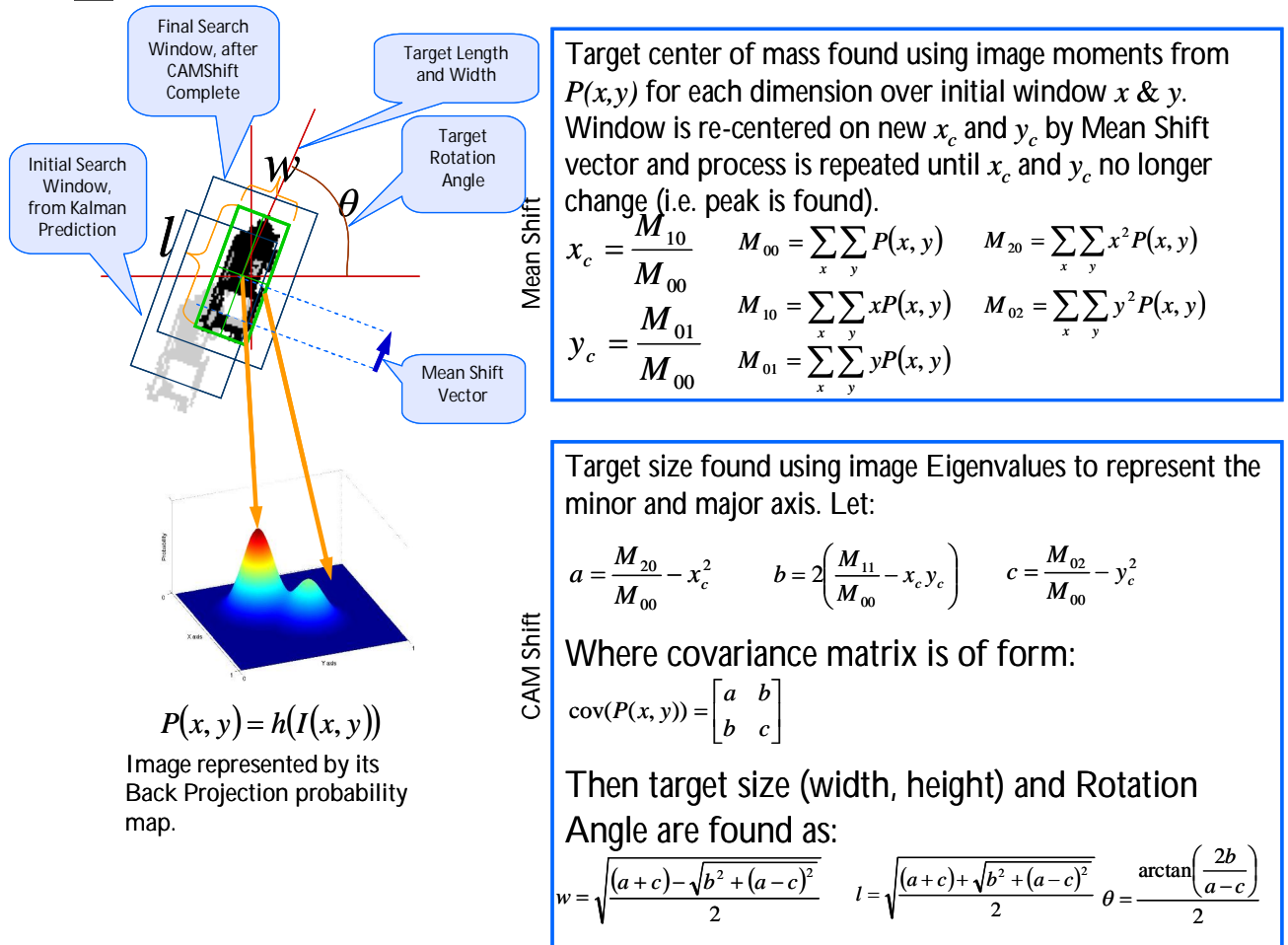


Figure 11: Overall Description of CAMShift

Kalman Filtering

The Kalman filter is an optimal estimator used by this project to predict where a target will be in future frames, based upon its movement in the past. The Kalman filter is a set of algorithms that implement a predictor-corrector type estimator that is optimal in the sense it minimizes the error estimate given input conditions.

This prediction takes into account any noise in the process itself or the ability to measure the process. The correction process uses the actual measurements of the system and incorporates the measurement noise to compute an optimum prediction for the next step. The net effect is a predictor that can remove noise from the system while accurately predicting the location of the target in future frames.

There are three Kalman filters used for this project, one to track the CAMShift area, one to track the target position from frame to frame (fast tracker), and one to track the target position through occlusions (slow or occlusion tracker). Each of these equations is 4x4, which allows position and velocity to be tracked. The Kalman filter has a useful feature in that it can provide the velocity of a system given just the position measurements, which is used by the area tracker to detect occlusions and the slow tracker to estimate object velocity.

Background - Process to be estimated

The system of equations used to compute the position and velocity of the X and Y coordinates are compactly represented as a discrete-time controlled process by the linear stochastic difference equation matrix form as:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

Where k is the step index, x_k is the state matrix, A is the transfer matrix and w_k is the inherent noise of the system, called the process noise expressed as the covariance matrix Q where $p(w_k) \approx N(0, Q_k)$. The term u_{k-1} represents an external control input. Since our targets are independently maneuvering, this term is zero and is not discussed further.

For this project we are assuming a constant acceleration model and the delta time step ($\Delta=1$) is one, so the state equation in matrix form is:

$$\begin{bmatrix} x_k \\ x'_k \\ y_k \\ y'_k \end{bmatrix} = \begin{bmatrix} 1 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_{k-1} \\ x'_{k-1} \\ y_{k-1} \\ y'_{k-1} \end{bmatrix} + w_k$$

The Process Noise and Measurement Noise covariance matrices for the Kalman filters are shown below with the numerical values following (using $m=3.0$ for the Measurement matrix and $p=8.0$ for the Process Noise matrix).

$$Q = \begin{bmatrix} \frac{2p^2\Delta^3}{3\tau} & \frac{p^2\Delta^2}{\tau} & 0 & 0 \\ \frac{p^2\Delta^2}{\tau} & \frac{2p^2\Delta}{\tau} & 0 & 0 \\ 0 & 0 & \frac{2p^2\Delta^3}{3\tau} & \frac{p^2\Delta^2}{\tau} \\ 0 & 0 & \frac{p^2\Delta^2}{\tau} & \frac{2p^2\Delta}{\tau} \end{bmatrix} \text{ or } Q = \begin{bmatrix} 1.422 & 2.133 & 0 & 0 \\ 2.133 & 4.267 & 0 & 0 \\ 0 & 0 & 1.422 & 2.133 \\ 0 & 0 & 2.133 & 4.267 \end{bmatrix}$$

The corresponding measurement equation is:

$$z_k = H_k x_k + v_k$$

The random variables v_k represent the noise inherent in the measurement process, called the measurement noise, expressed as the covariance matrix R where $p(v_k) \approx N(0, R_k)$.

For this project the measurement noise covariance;

$$R = \begin{bmatrix} m^2 & 0 \\ 0 & m^2 \end{bmatrix} \text{ or } R = \begin{bmatrix} 9 & 0 \\ 0 & 9 \end{bmatrix}$$

The slow Kalman filter is tuned to weigh the prediction more heavily than the measurement due to the occlusion, so the measurement parameter is increased as follows:

$$R = \begin{bmatrix} 30^2 & 0 \\ 0 & 30^2 \end{bmatrix}$$

Both Q and R are assumed to be independent of each other, white and with a 0 mean and normal probability distribution. In practice the process and noise covariance matrices might change with each step, however we assume they are constant for this project as shown above. The matrix H is called the measurement matrix and is simply expressed as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Where the 1's represent the x and y position measurements. Since we are not measuring velocity directly, 0's are located in those positions.

Kalman Filter Equation

The Kalman filter is used to estimate the state matrix x_k given prior values of the state, a transfer matrix, and noise estimates. The goal of the Kalman filter equation is to compute the *a posteriori* state estimate (\hat{x}_k) as a liner combination of an *a priori* estimate (\hat{x}_k^-) plus a weighted

difference between the actual measurement (z_k) and a measurement prediction ($H\hat{x}_k^-$) that minimizes difference between the past estimated and actual measurements. The Kalman equation is shown below.

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H\hat{x}_k^-)$$

The process noise covariance matrix Q and measurement noise covariance matrix R are assumed to be constant throughout the process. The $(z_k - H\hat{x}_k^-)$ term represents the *innovation* or residual and reflects the difference between the predicted measurement $H\hat{x}_k^-$ and actual measurement z_k . If the residual is zero, it means the prediction and measurement are in complete agreement. The key computation is for K_k , which represents the Kalman Gain and is discussed below.

The Kalman filter is used to estimate the state matrix x_k given prior values of the state, a transfer matrix, and noise estimates. The goal of the Kalman equation is to compute the *a posteriori* state estimate (\hat{x}_k) as a linear combination of an *a priori* estimate (\hat{x}_k^-) plus a weighted factor called the *innovation* or *residual* and it represents the difference between the predicted $H\hat{x}_k^-$ measurement and the actual measurement (z_k) and a measurement prediction ($H\hat{x}_k^-$) that minimizes difference between the past estimated and actual measurements. Below summarizes the equation.

The matrix K is called the Kalman *gain*, or *blending factor*, and is used as a factor to minimize the *a posteriori* error covariance P_k^- , which is the end goal of the process. The most common form of the Kalman equation that minimizes P_k^- is given by.

$$K_k = P_k^- H^T (H P_k^- H^T + R_k)^{-1}$$

Another way to look at the weighting K_k is as the measurement error covariance R_k approaches zero, the actual measurement z_k is “trusted” more and the predicted measurement $H\hat{x}_k^-$ is trusted less. Conversely, as the *a priori* estimate error covariance P_k^- approaches zero, the actual measurement z_k is trusted less and the predicted measurement $H\hat{x}_k^-$ is trusted more. The key calculation of the Kalman update step is to compute the value of K such that the *a posteriori* error is minimized.

Discrete Kalman Filter Equation

The Kalman filter estimates a process by implementing a type of feedback control, i.e. the filter estimates the process state at some point in time, and then obtains feedback in the form of

measurements and measurement noise and updates the estimate. The Kalman filter falls into two discrete time update equations: the prediction step, where the next process state (*a priori*) is predicted; and the correction step where the actual measurements are used to compute an improved *a posteriori* estimate.

Figure 12 below shows the implementation as used in OpenCV (Welch & Bishop, p. 24, 2001). The Predict step is called prior to the CAMShift operation to place the search window in the predicted location, and the Correct step is called following the CAMShift operation, the Correct step is called to feedback the actual measurements and adjust the Kalman state for the next prediction.

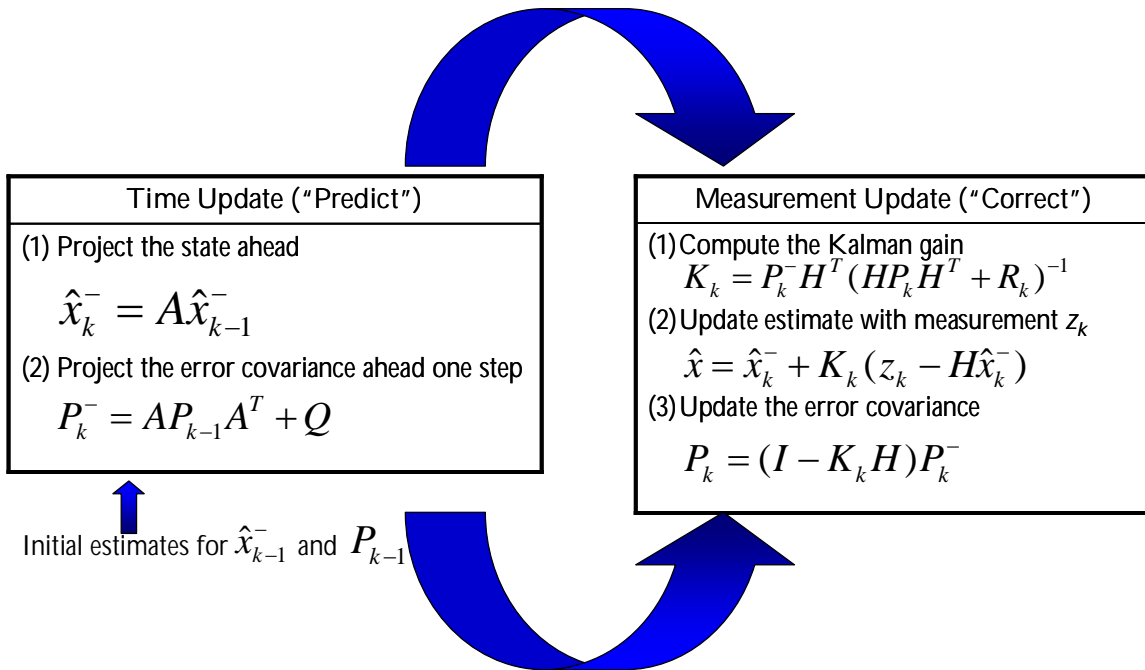


Figure 12: Kalman Predict/Correct Cycle

Results and Discussion

Kalman Filter Tuning

The fast Kalman filter, used to predict the position of the tracker from frame to frame, is subject to both process (inherent) and measurement noise. Adjusting either of these noise estimates impacts the noise covariance matrices used by the Kalman filter and impacts how it performs for a given environment. To tune the Measurement and Process Noise parameters, a series of tests were run on each of the three scenarios (Wooden Track - Side View, Wooden Track - Top View, and Cement Track - Top View). Each of the tests were automated by scripting the CAMShift program and having it run three times for each of the combinations of Measurement Noise Standard Deviation, Process Noise Standard Deviation, video file name, and video update rates. The video update rate was simulated by only processing every N_{th} frame to simulate a slower capture rate ($N=1, 2, \text{ or } 3$). The two following graphs Figure 13 & 14: CAMShift Convergence Iterations Mean and Kalman Tracker Accuracy show the result of averaging all of the data for the three video scenarios at the 30 FPS case. The data shows the average number of CAMShift convergence cycles and average Kalman accuracy prediction (non-occlusion case) for varying Process Noise and Measurement Noise coefficients. The value of Measurement Noise = 3 and Process Noise = 8 is an optimum case across the three scenarios. These values were then used as the default for subsequent testing. Note in the CAMShift graph, the Measurement noise of 0 indicates that no Kalman filtering was used, and therefore the prediction accuracy of using the prior CAMShift position is much less than the optimum Kalman filtering case.

The process noise and the measurement noise values that were chosen represent the optimum relationship between the measured value and the predicted value. The process noise is the inherent noise in the movement of the object being tracked, so when there is a high process noise, i.e. the motion of the object is much more random, the measurement of the tracker is going to be trusted more over the prediction due to the random and less predictable nature of the object. The measurement noise is the noise of the CAMShift function, so when there is a high measurement noise, i.e. when the tracker's ability is lacking due to a glare, occlusions, or poor picture quality, the prediction will be trusted more due to the unreliability of how the object is being measured. The process noise and measurement noise are independent of each other, however, so one does not necessarily have to go down if the other goes up, and vice versa. Due to this, there was an optimum point at which the effect of the two values was optimized, which was where the measurement noise

was 3 and the process noise was 8. At this point, the number of convergence cycles and the error of the Kalman filter were lowest, so the algorithm was optimized with those two values. These values were found consistently through the measurement of all the process noise and measurement values for all of the different videos, typically only varying 1 to 2 from the chosen values. The values were chosen based on an average of all the videos.

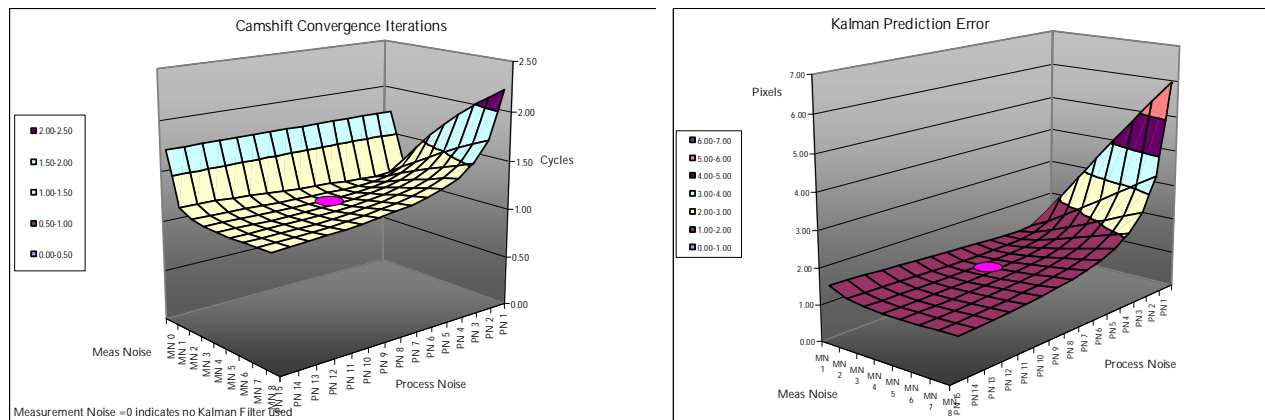


Figure 13 & 14: CAMShift Convergence Iterations Mean and Kalman Tracker Accuracy

Kalman Accuracy and Kalman Convergence Cycle Reduction

The fast Kalman filter proved to be very accurate in placing the CAMShift window into the correct location each frame. Correct placement of the window not only reduces the chance of the wrong target being tracked, but it also reduces the computational load of the CAMShift algorithm to correctly center on the target. Figure 15 and Figure 16 below show results from the Wooden track case with no occlusions. The Cement track case provides similar results. Figure 15 shows the accuracy of the fast Kalman filter in predicting the location of the center of each CAMShift window. As shown, it is generally within a pixel of the correct placement over the complete course vs. 5 pixels, which is the typical velocity of the target from frame to frame.

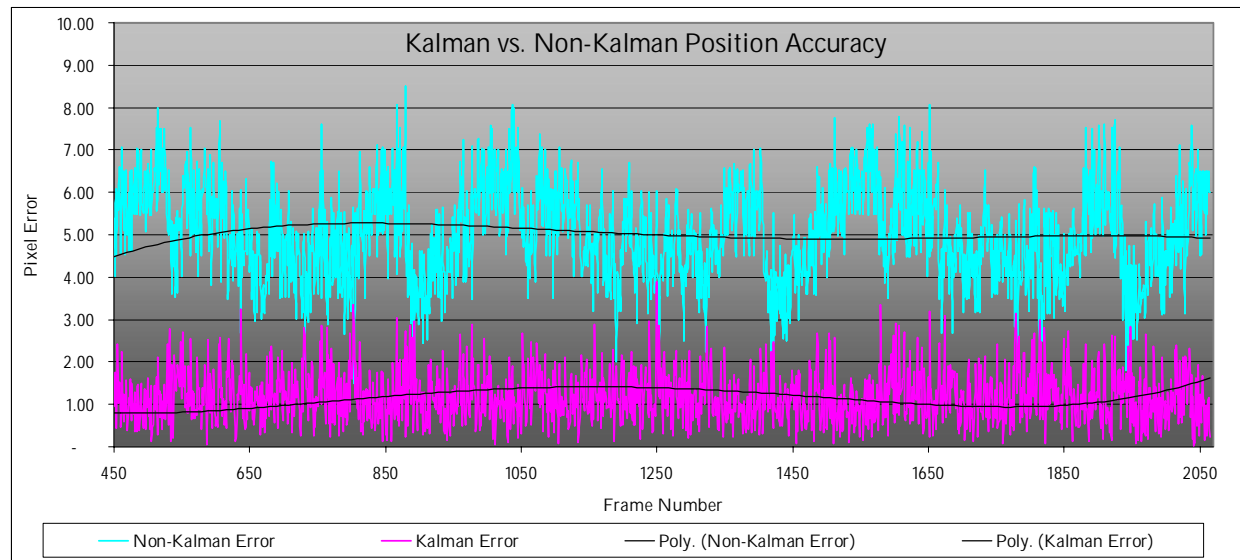


Figure 15: CAMShift Window Placement Accuracy for Kalman and non-Kalman case

Figure 16 shows the net improvement in the total number of convergence cycles used by the CAMShift for the combination of the Wood track and Cement track (no-occlusions). Without the Kalman filter a total of 7,505 cycles were required, and with it a total of 4,908, a processing reduction of 35%. Since the CAMShift algorithm is computationally complex, this is a significant savings in computing resources.

This improved accuracy of the tracker with the Kalman filter is advantageous for two reasons: 1) It results in a reduction in the number of convergence cycles for the CAMShift tracker thus reducing computation; and 2) Reduces the chances of non-target pixels are included in the following CAMShift operation. This improved accuracy was found in multiple trials for each of the three videos.

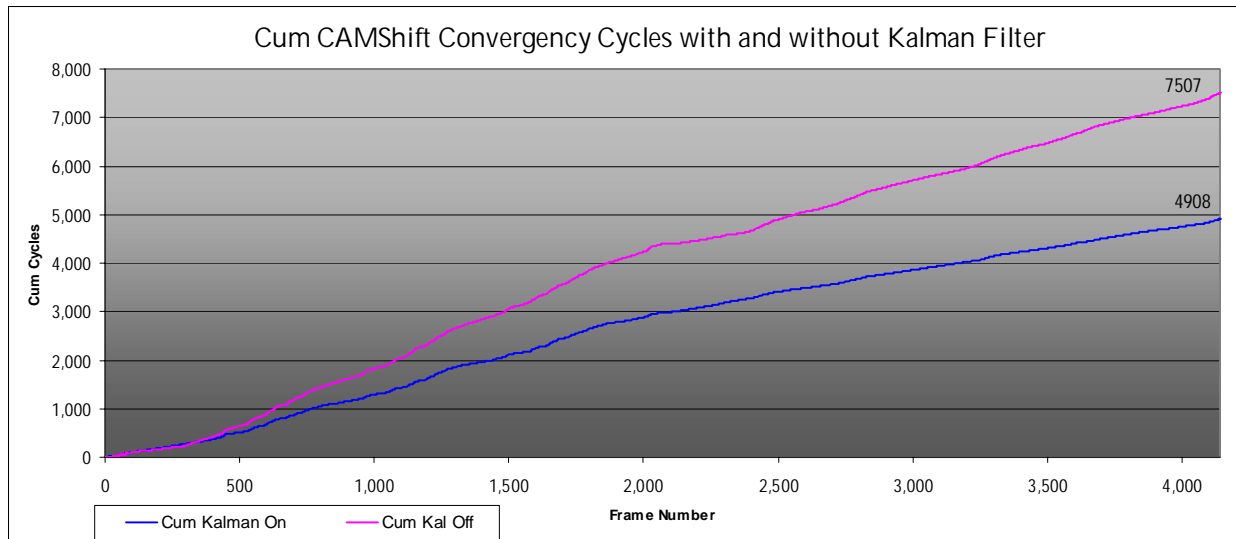


Figure 16: Cumulative Convergence Cycles for Kalman and non-Kalman case

Occlusion Detection

The most challenging aspect of this project is to detect then maintain a track on an object as it goes through, and comes back out of, an occlusion. Initially, it was thought a single Kalman filter could be used for both the non-occlusion update and the occlusion prediction. However, it was discovered that as the target was being occluded, it shrunk in size. As it shrunk, the center of mass of the object was only moving at half of the speed as the object itself since the front of the object was no longer moving (since it was being occluded by a fixed object). Therefore, the fast Kalman filter started slowing down as the mass center started slowing down. Thus, it would not properly pick up the target as it exited the occlusion. Also, depending on the aspect angle of the target as it entered the occlusion, the velocity vector (angle) would start to rotate due to the target size being “squeezed” down on the side. This would cause the target to drift off to the side when the fast filter was free run during the occlusion. Two steps were taken to correct this condition. 1) Early detection logic was added to detect an occlusion while it was occurring, rather than waiting for the object to disappear; 2) An additional slower reacting Kalman filter was added to smooth out the rapid decrease in velocity.

Figure 17 below shows how the occlusion detection logic is used to detect an occlusion before the target disappears under the occlusion completely. A ratio of the rate of change of the area divided by the area itself is computed (using Kalman filter data). If this ratio is less than -25%, then the target is quickly decreasing in size and thus considered occluded and about to be lost completely. In addition the CAMShift area, or total value of the probability distribution in the returned search window, is tracked. If this area drops suddenly, to less than 50% of the last known value, then

occlusion is noted. This property is also used to allow occlusion to end, in that the CAMShift area must be greater than the last 50% value of a good track. This property allows the CAMShift tracker to briefly latch onto to phantom artifacts during occlusion but successfully reacquire the object when it is no longer occluded. The graph below shows the ratio values as the target enters and leaves the four occlusions.

The values of ACQ and the occlusion detection were measured multiple times for each of the videos, and each time gave similar results.

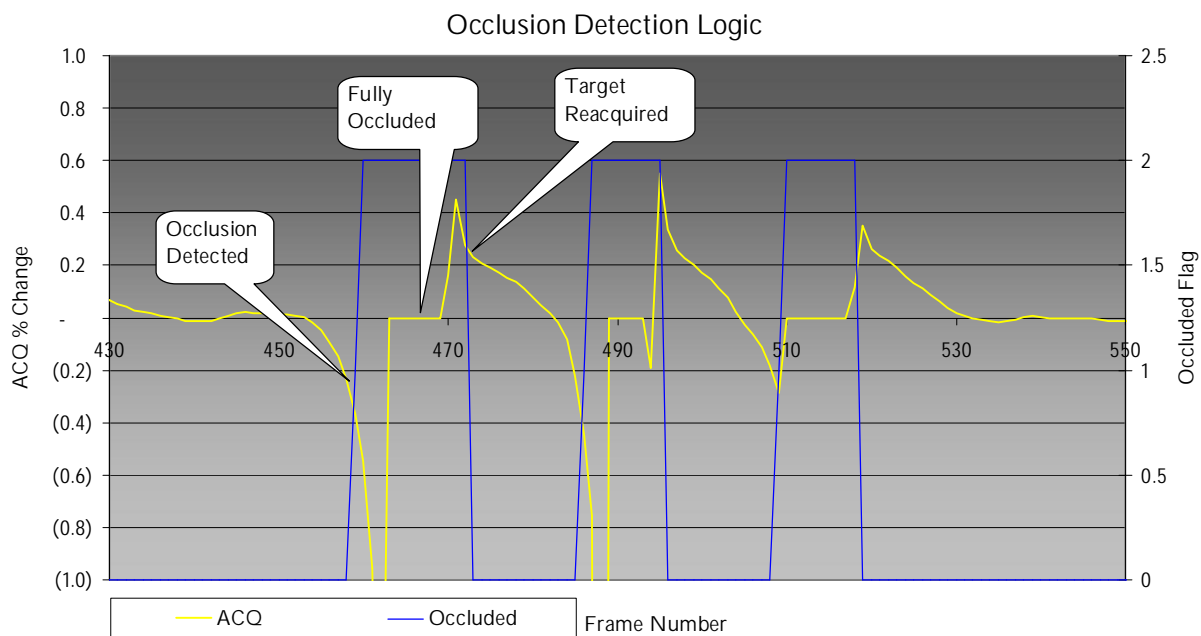


Figure 17: Occlusion Detection Logic based on CAMShift Area Rate of Change

Along with the occlusion detection logic, a slow Kalman filter was added that greatly smoothed out the both the velocity and angle of the target as it entered the occlusion. Figure 18 below shows the three synthetic occlusions on the Wooden Track – Top View, frames 430 to 550. The blue lines represent the occlusion detection logic and the red and yellow lines represent the actual and slow Kalman filter velocities respectively. The actual velocity was capture during a trial without the occlusions. Note that the slow Kalman filter is smoothing out the actual velocity. However, as the target starts entering the occlusion, the Kalman filter velocity decreases rapidly and as soon as the occlusion is detected, the filters are set into a free-running state where their measurement values are updated with the prediction values.

If the occlusion was accurately detected earlier, the Kalman filter would have a more accurate reading of the velocity. However, very early detection of the occlusion is a very difficult problem to solve and would need much more sophisticated algorithms than used here.

The slow filter recovers from the rapid decrease in speed and continues to run through the occlusion at a constant speed similar to the speed of the object before the occlusion. Note when the target is reacquired, the filter values jump then settle down as they are updated to the correct positions.

Since the slow Kalman is able to maintain the velocity through the occlusion, it shows that it is able to efficiently track the object through the occlusion. The velocities of each of the Kalman filters were measured multiple times for each of the videos, and there were similar results for each reiteration.

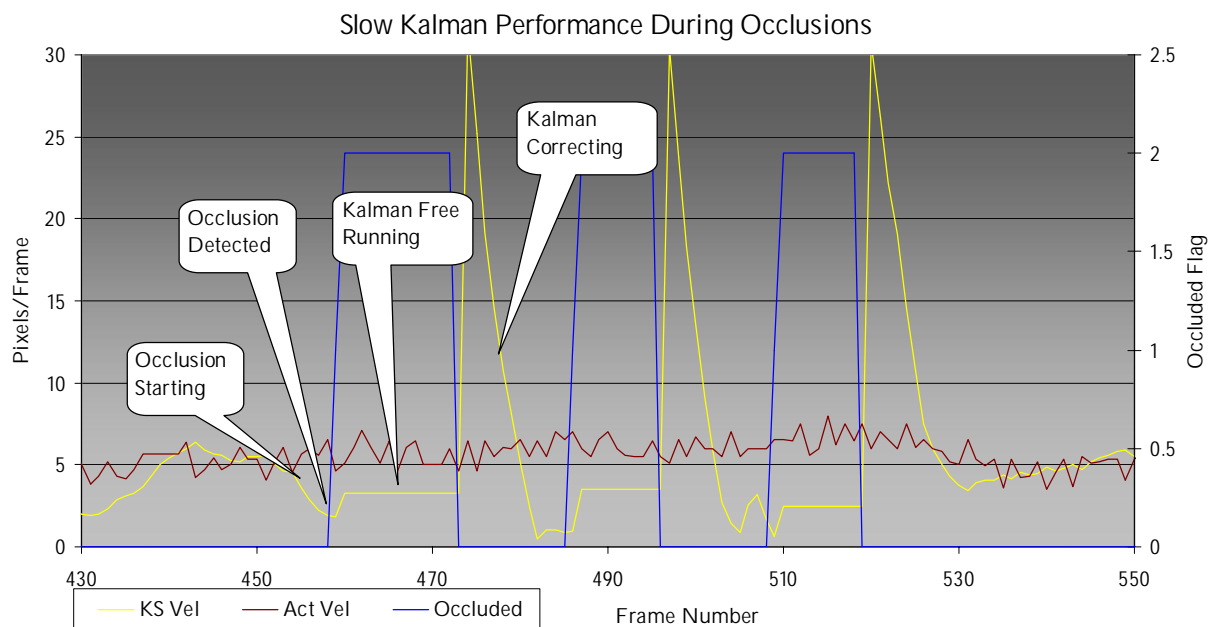


Figure 18: Slow Kalman Filter Performance and Occlusion Detection

Figure 19: Occlusion Track Angles below shows the accuracy of the slow Kalman filter velocity angles as they enter the occlusions for the most complex case (Wooden Track – Top View, expanded to show frames 300 to 840). The purple line shows the velocity angles for the case without any occlusions, and the yellow line shows the angles with the occlusions. The goal is to have the angles set properly as the target enters the occlusions so that when the slow Kalman estimator free-runs during the occlusion, the prediction box will successfully reacquire the target as it exits the occlusion. The results show a high accuracy between the two cases. Note the angles for the occlusion case remain fixed during the occlusion since the tracker is free-running.

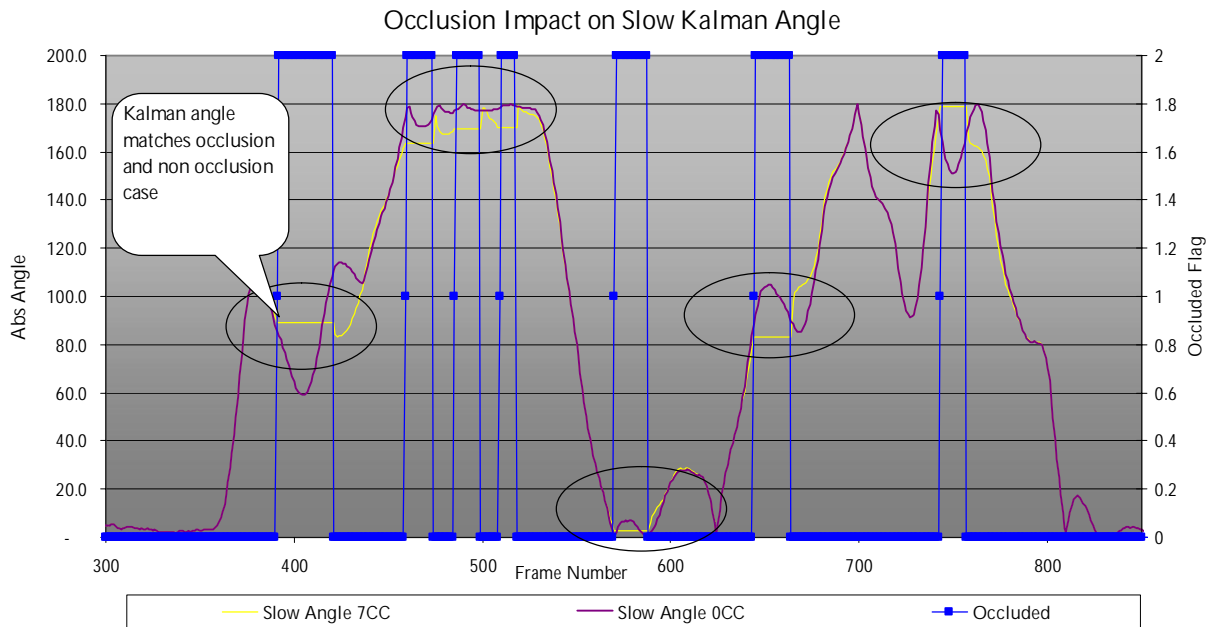


Figure 19: Occlusion Track Angles

Figure 20 below shows a video frame sequence of a target as it enters one of the artificial occlusions on the Wooden track. The green ellipse indicates a good track from the CAMShift tracker and as shown in frame #452. Frame #459 shows the target as it is being detected by the frame occlusion logic (due to rapid size decrease). The CAMShift tracker is still tracking it, since a red ellipse is shown; however, the slow Kalman filter has been engaged. The yellow cross-hair/tail indicates the position and direction of the slow tracker. Frame #470 shows the target as it moves under the occlusion and the search window being moved and resized based on the prediction of where the target will emerge. Frame #473 shows the target being reacquired. Frames #476 & #493 repeat the process of the target being covered by the second occlusion and the Kalman filter tracking it.

Figure 21: Occlusion Frame Sequence – Cement Track shows a similar scenario using the cement track. In this case the target is smaller and closer in color to the floor, which is a more challenging case. However, as shown the target is successfully tracked through the occlusion.

This visual representation of the algorithm's effectiveness show that the methods described above for the detection and tracking of objects through occlusions are robust and useful. This process was repeated multiple times for all three videos, as well as in real-time, and the algorithm works robustly for all of the used tests, showing the overall effectiveness of this algorithm for a majority of situations.

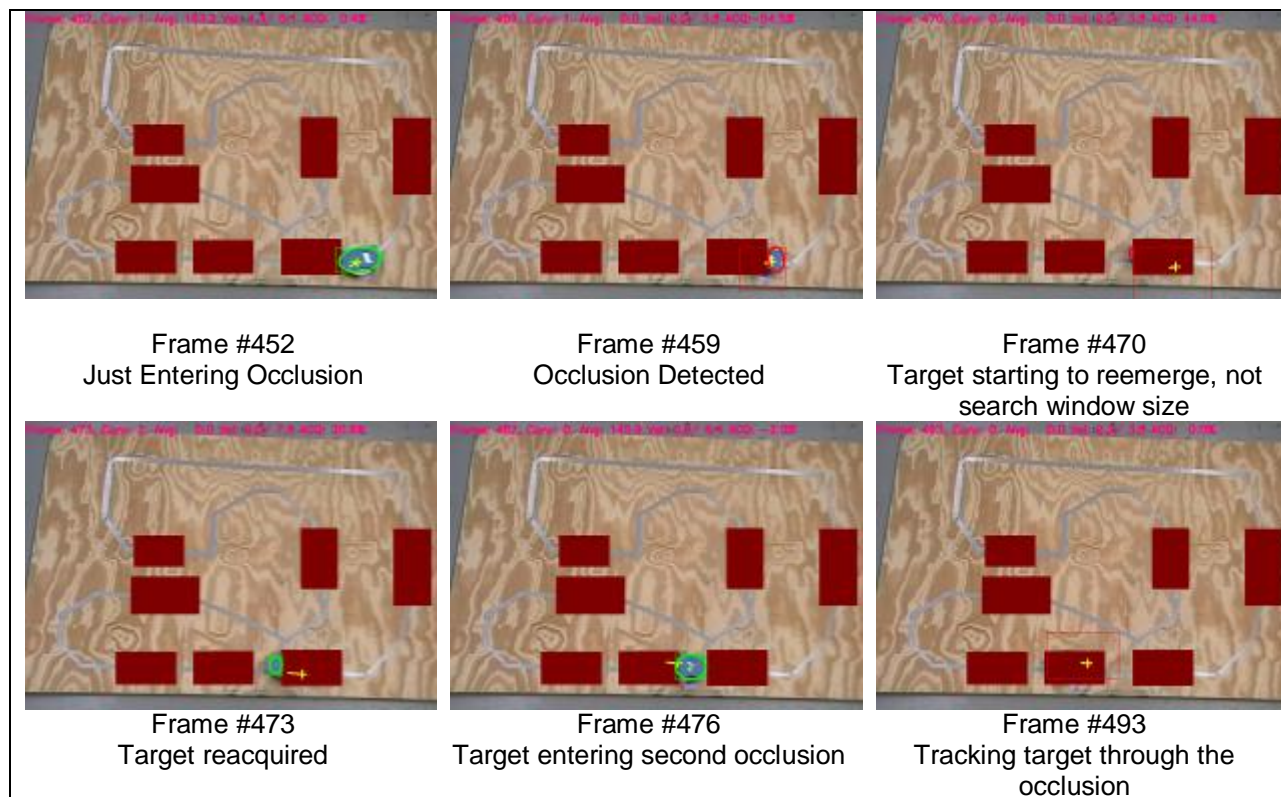


Figure 20: Occlusion Frame Sequence – Wood Track Top View

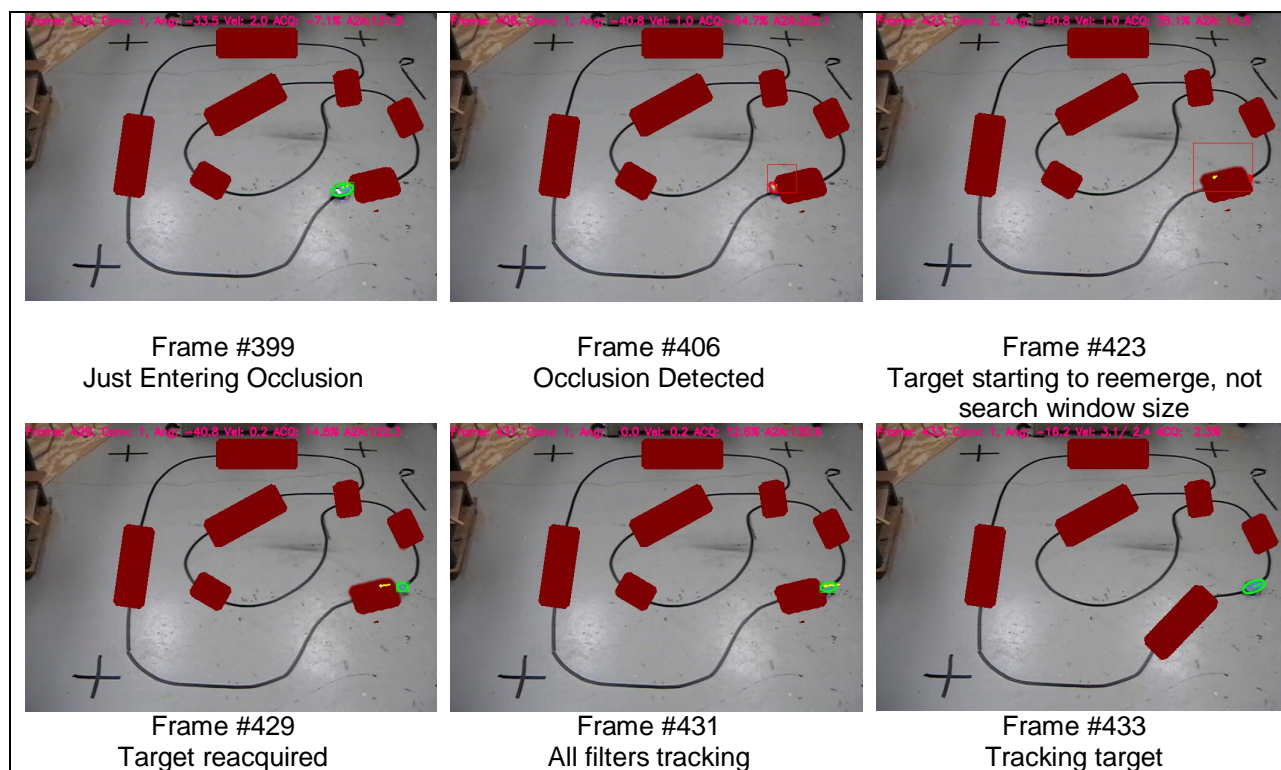


Figure 21: Occlusion Frame Sequence – Cement Track

Improvements due to Saturation and Local Binary Patterns

The results from the prior section were achieved using Hue as the primary image component for the back projection computation. The video sequences are non-complex and work well for testing, measuring and tuning the algorithm. However, the scientist wanted to use the completed algorithm on real life subjects such as moving objects if front of the camera quickly, tracking pedestrians or other complex objects. Continued research, as described in the first section, indicated that tracking another component of the image color, such as saturation, and image texture would help discriminate the target against complex backgrounds. After researching texture identification and classification, the idea of using Local Binary Patterns was tried along with using the object saturation along with hue. To test the enhanced algorithm, a number of video sequences were used. One included myself walking behind an occlusion a number of times. A sequence is shown below.

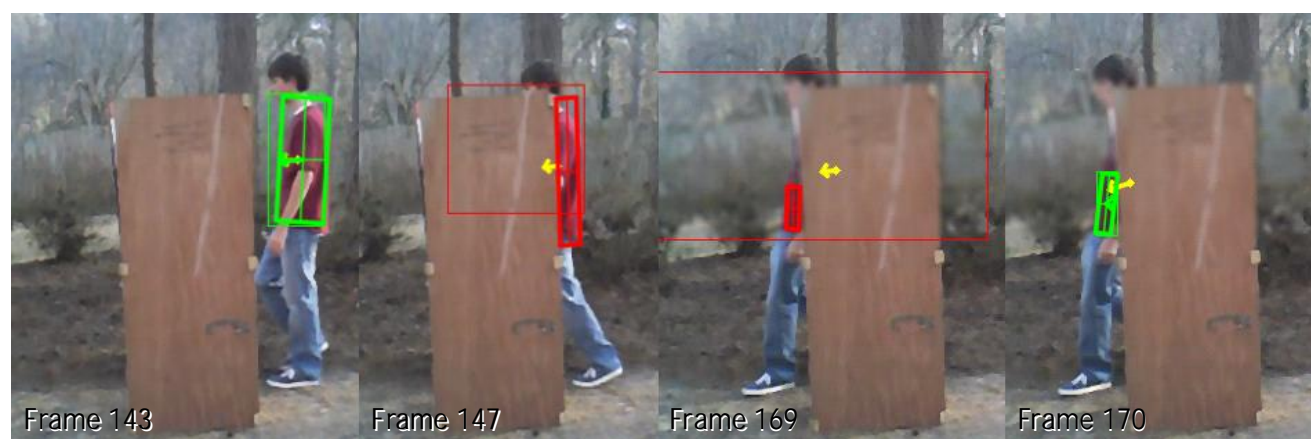


Figure 22: Occlusion Frame Sequence – The Author Walking behind and Occlusion

This above video did not prove to be much of a challenge, so additional videos were considered. The scientist thought of an idea while watching short track speed skating on the winter Olympics. I would like to take a short track video and track a single skater around the track for the entire race. This proved to be challenging, however the tracker is able to successfully perform this function on a number of video sequences.

To show the improvement adding the saturation and LBP functions, a video sequence is shown Figure 23: Improvements Due to Saturation + LBP that on the left shows the same frame from three different implementations of the algorithm. The right hand shows three other identical frame sequences as well. The top row shows the original algorithm tracking a skater (the one in the red pants) across an almost identical colored background. Because the hues are so similar, the entire

background red banner and the skater are tracked by the CAMShift tracker. The result of this is when the skater leaves the banner area, the tracker stays locked on the banner.

The second row shows the improvement due to the addition of the saturation channel to the histogram and back projection algorithms. On the left hand side you can see the CAMShift returned track box is smaller than on the top row, which denotes the tracker is focused more on the skater and less on the banner. However, there is still some of the banner being tracked. This causes the tracker to momentarily lose the skater as get farther away from the banner area. On the right hand side, second row, there isn't much of an improvement made by the addition of the saturation channel. The result is adding saturation is not enough to maintain track of the skater against very similar colored and density backgrounds

The third row shows improvement due to the LBP_g^{ri36} to the histogram and back projection algorithms. In both rows there is a clear improvement in tracking and the complete hue + saturation + LBP method works very well in tracking the skater through the background that is of a nearly identical color as the skaters red pants. It is not obvious to the eye, but there is enough difference in texture that the back projection algorithm is able to create a probability density function that correctly isolates the skater from the background.

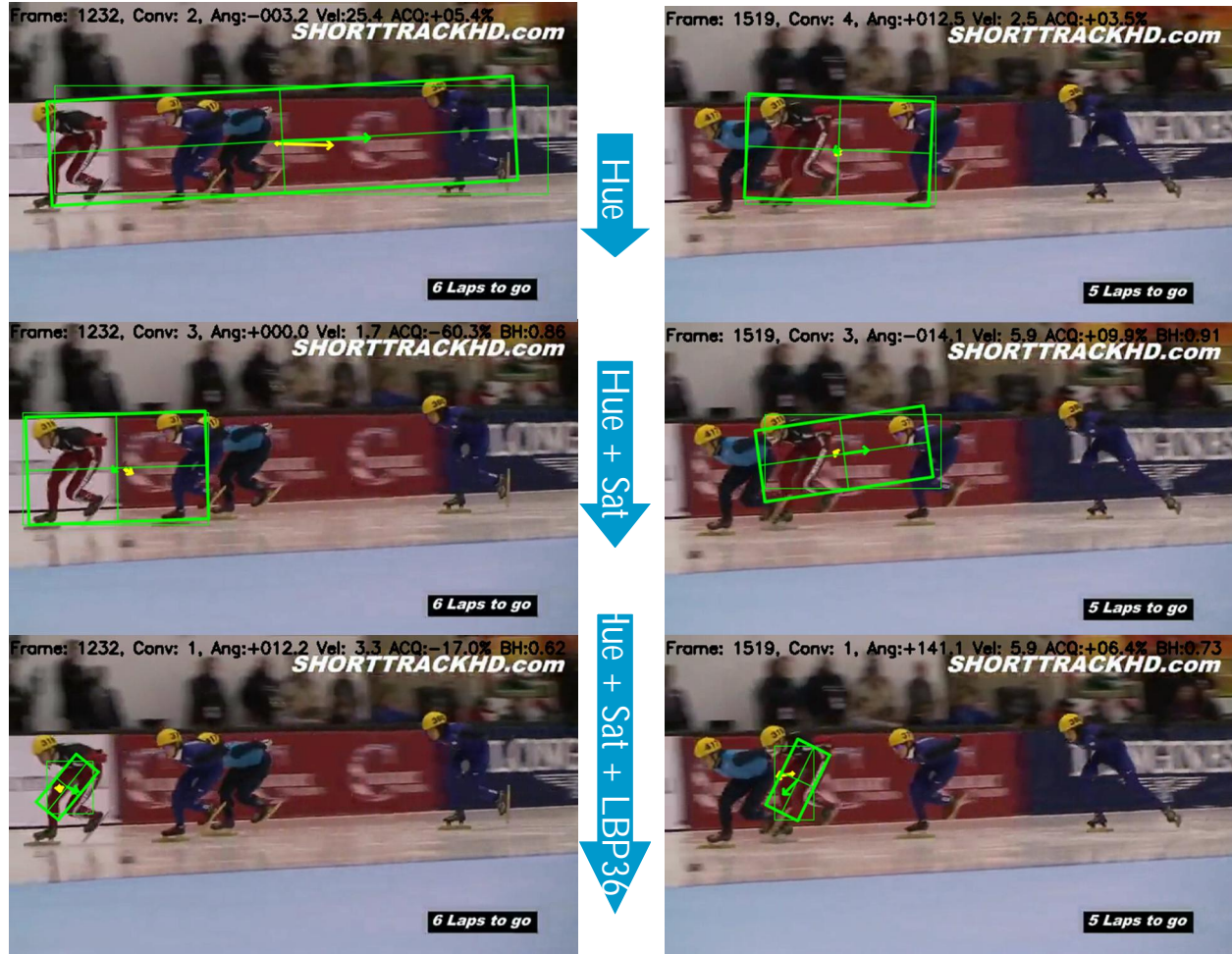


Figure 23: Improvements Due to Saturation + LBP

To highlight the improvements made by the addition of saturation and LBP channels to the back projection process, Figure 24: Back Projection – Improvements Due to Saturation + LBP shows the corresponding back projection images for the ones shown in Figure 23. The improvement on how the skater is isolated from the similar colored background is clearly evident. There isn't much improvement from the Hue to Hue+Sat case, however with the addition of the LBP the skater is clearly highlighted from the background which allows the CAMShift function to track the skater through the color occlusion.

Conclusion

The results show the finalized code is able to efficiently track an object as it moves, including through occlusions. The resulting project is able to maintain track of an object for as long as needed, even with the object is occluded by objects and similar colors. The project goal was to design and implement a combination of the Back Projection, CAMShift, and Kalman estimator algorithms to track objects in a video sequence or in real time through multiple occlusions, and to reacquire the objects when they reappear. This goal was met, as the combination of these functions into a single algorithm created an efficient method of tracking through occlusions. The Kalman filter enhances the CAMShift function centering the window on the object, thus reducing the number of cycles needed for convergence and maintaining a track of the invisible object as it moves through occlusions. The method of detecting and tracking through occlusions proves to be very effective and works well for a majority of situations. The use of color and texture to isolate the object proved successful and the implementation met its goals of running in real time. Overall, the algorithm created by this project is both a useful and powerful method of tracking objects, especially when they become occluded.

During the research and experimentation, there were a variety of challenges and errors that came about from the project, with some corrected, but others not completely resolved. Some of the more challenging issues included the early detection of the occlusion, the speed and direction of the Kalman filter during occlusion, and the reacquiring of the object after the occlusion. However, some tracking errors, including the movement of the center of mass and thus the slowing down of the Kalman filter velocity, similarities in hue between the background and the target, and the direction of the object when entering the occlusion, were not completely resolved, but were handled successfully in the software.

The program also works for live video, or web-cam use, and it has proven to track objects moved by the authors hand very well.

The initial implementation of the CAMShift tracker was able to recognize the objects based on hue only, and it become confused when the background and the object have similar hues, as it is unable to differentiate the two. This was seen during the video with the cement floor, as there was a point where the light gray of the cement was very similar to the blue of the robot, so the tracker became confused and began to track the floor once the object became occluded. That was initial overcome by automatically selecting the Saturation and Luminance thresholds and by implementing a weighted histogram that reduces the impact of common hues and enhances the impact of the

unique object hues for the CAMShift tracker. Later, the addition of Local Binary Patterns to further isolate the object greatly increased the capability to track similar colors. The addition of the dynamic updating of the target histogram to account for lighting changes improved the track for long sequences or sequences with varying lighting conditions.

A final issue is the aspect angle of the object, the angle it has when entering the occlusion vs. when it exits the occlusion. If the object moves around a curve and then enters the occlusion, the aspect angle is different from the angle of the track, so the search box grows in the incorrect direction. This can be overcome by growing the search box in more directions than the angle it enters from, but this increases the possibility of locating something of a similar color that is not the object. To overcome this issue, acceleration was added to the slow Kalman filter. This allowed the target to maintain an angular velocity as it moves through the occlusions. However, this method did not improve the performance, but rather made it worse. This model caused the target to move in the opposite direction in some cases and improved it in others. The filter was restored back to the constant acceleration case.

This algorithm has a variety of practical applications in the military, such as information gathering, reconnaissance, and better methods of tracking targets. It also can be used to increase the possibilities of robotic uses, such as use in factories to be able to locate and track specific materials, traffic and crowd control, security purposes, and expansion of artificial intelligence to increase the visual ability of autonomous robots.

Continued Research

Immediate improvements to the project include the ability to track multiple objects,. Also improvements to the template matching to use additional techniques such as neural networks and other image recognition techniques to identify objects both initially and following an occlusion. The occlusion detection and target tracking through occlusion has room for improvement as well, with the goal of detecting the occlusion just as it happens. Methods of optical flow may be researched and applied to improve this capability. Another area of research is to have the camera track the object by mounting onto a servo platform that can allow it to move around and track in a wider range.

Long term research for this project will include using remote cameras, possibly multiple cameras, that can be maneuvered or maneuver themselves to track and identify objects. The long term goal is to have a swarm of robots, each with a camera, working together to discover and track objects.

References/Literature Cited

- [1] Agarwal, Prakhar, et al. "TARGET TRACKING: IMPLEMENTING THE KALMAN FILTER." Web. 20 Sept. 2009.
- [2] Artner, "A Comparison of Mean Shift Tracking Methods", Digital Media, Upper Austria University of Applied Sciences, 2008
- [3] Ahmed, Javed. "ADAPTIVE EDGE-ENHANCED CORRELATION BASED ROBUST AND REAL-TIME VISUAL TRACKING FRAMEWORK AND ITS DEPLOYMENT IN MACHINE VISION SYSTEMS." Thesis. National University of Sciences and Technology, 2008. Bibliography of PhD Thesis. Web. 20 Sept. 2009.
- [4] Allen, J. et al. "Object Tracking using CamShift Algorithm and Multiple Quantized Feature Spaces" Sydney: Proceedings of the Pan-Sydney Area Workshop on Advances in Pattern Recognition. 2007
- [5] Berry, Greg. "Blob-tracking software." IFP - Image Formation and Processing. 1998. Web. <<http://www.ifp.illinois.edu/~berry/thesis/node27.html>>.
- [6] Bradski, Gary, and Adrian Kaehler. Learning OpenCV. O'Reilly Media, 2008. Print.
- [7] Bradski, Gary. "Computer Vision Face Tracking for Use in a Perceptual User Interface." Intel Technology Journal" Web Q2, 1998.
- [8] Collins, Robert. "Online Learning for Tracking." Lecture. VLPR Summer School. Beijing, China. Pennsylvania State University, 25 July 2009. Web. 20 Sept. 2009.
- [9] D. Comaniciu "Mean Shift: A Robust Approach Toward Feature Space Analysis," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 24, No. 5, May 2002.
- [10] D. Comaniciu and V. Ramesh. "Kernel-Based Object Tracking," IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head SC. Web 2000.
- [11] D. Comaniciu and V. Ramesh. "Mean Shift Optimal Prediction for Efficient Object Tracking," Proc. Int'l Conf. Image Processing, vol. III, pp. 70-73, 2000. 20 Sept. 2009.
- [12] Fukunaga, K. and Hostetler, L. "The estimation of the gradient of a density function, with applications in pattern recognition", IEEE Transactions for Information Theory. Volume 21, Issue 1, page(s) 32-40 : Jan 1975
- [13] Mäenpää T & Pietikäinen M, "Texture Analysis with Local Binary Patterns", Chen CH & Wang PSP (eds) Handbook of Pattern Recognition and Computer Vision, 3rd ed, World Scientific, 197-216, 2005
- [14] Ojala, Timo et al, "Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns", Computer Vision, ECCV 2000 Proceedings, Lecture Notes in Computer Science 1842, Springer, 404-420, 2000
- [15] Li, Rong X. and Jilkov, Vesselin P.. " A Survey of Maneuvering Target Tracking: Dynamic Models" Proceedings of SPIE Conference on Signal and Data Processing of Small Targets, 4048-22, April 2000
- [16] Stolkin, R. et al. "An adaptive Background Model for CamShift Tracking with a Moving Camera", Proceedings of the 6th International Conference on Advances in Pattern Recognition. 2007

- [17] Welch, Greg, and Gary Bishop. "An Introduction to the Kalman Filter." Reading. SIGGRAPH 2001. Los Angeles, CA. An Introduction to the Kalman Filter. UNC at Chapel Hill. Web. 20 Sept. 2009. <<http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html>>.
- [18] Y.M. Kim. "Object tracking in video sequence." <http://www.stanford.edu/~jinhae/CS229/report.pdf>. 20 Sept. 2009.
- [19] Yilmaz, Alper, Omar Javed, and Mubarak Shah. "Object Tracking: A Survey." ACM Computing Surveys 38.4 (2006). ACM Computing Surveys, Dec. 2006. Web. 20 Sept. 2009.
- [20] Z. Zhu, Q. Ji, K. Fujimura, and K. Lee. "Combining Kalman Filtering and Mean Shift for Real Time Eye Tracking under Active IR Illumination," Proc. Int'l Conf. Pattern Recognition, Aug. 2002. 20 Sept. 2009.
- [21] Zhang, Chunrong, Yuansong Qiao, Enda Fallon, and Chiangqiao Xu. "An improved CamShift algorithm for target tracking in video surveillance." Reading. 9th IT&T Conference. Dublin, Ireland. Dublin Institute of Technology, 2009. Web. 20 Sept. 2009.

Acknowledgements

I would like to give a formal acknowledgement to Tom Wilkason, my father, who taught me all of the difficult concepts, and who was essential in the programming of the code. He guided me through the process of the research and experimentation, and helped me with anything that was difficult.

I would also like to thank my Mother, Laura Wilkason, for checking my report and poster for grammatical errors.

Appendix A - Software Listing