The Pennsylvania State University

The Graduate School

College of Engineering

REAL TIME OBJECT TRACKING ON ACTIVE PAN TILT ZOOM CAMERA USING CMT

A Thesis in

Computer Science & Engineering

by

Kameran Davis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

May 2015

Abstract:

Object tracking is a fast growing topic in computer vision. Object tracking is especially useful in terms of unmanned and manned aerial, ground, and underwater vehicles, making inroads to replace manned vehicles for purposes such as videography, structural monitoring, surveillance, and aerial mapping. Other applications include motion-based detectors as well as human-computer interaction. In this thesis, a feedback-based solution to automatically track a selected target is proposed and implemented, based on object motion in different parts of the scene and experimentation is performed to illustrate fulfillment of the Pan Tilt Zoom operations of the camera while tracking test subjects in real-time. The approach uses Consensus Based Matching and Tracking of Keypoints to track the centroids of the objects. After each frame, the algorithm updates the location of the object centroid relative to the center of the camera's point of view. With the proposed real-time tracking system, this location is then used to calculate what direction and by how many degrees does the camera need to move so that the object of interest is contained with the camera's center of view. This solution can be even be extended to function on multiple cameras tracking a single object, allowing to factor in other elements such as depth perception, or even the expanded to function as part of a fully autonomous localization and navigation systems for unmanned vehicles and robots.

Contents:

List of Figures:

List of Tables:

List of Abbreviations:

2D	Two-Dimensional	12
3D	Three-Dimensional	11
BRISK	Binary Robust Invariant Scalable Keypoints	4
CAMShift	Continuously Adaptive Mean Shift	11
CMT	Consensus Based Matching and Tracking of Keypoints	20
CPU	Central Processing Unit	35
CV	Computer Vision	5
FAST	Features from Accelerated Segment Tests	4
FLANN	Fast Library for Approximate Nearest Neighbors	19
FOV	Field of View	37
FT	Fragments-Based Tracking	22
GFTT	Good Features to Track	18
GLOH	Gradient Location Orientation Histogram	7
GMM	Gaussian Mixture Model	12
HT	HoughTrack	22
KD	K-Dimensional	7
LM	LearnMatch	22
MAP	Maximum A Posteriori	15
MRF	Markov Random Filter	7
MS	Mean Shift	7
PTZ	Pan-Tilt-Zoom	1
SB	Semi-Supervised Online Boosting	22
SIFT	Scale-Invariant Feature Transform	4
STRUCK	Structured Output Training	22

Acknowledgements:

I first give praise to my Lord and Savior Jesus Christ for being with me through all of my ups and downs of life. Without him, I would not be where I am today.

I would like to acknowledge The Pennsylvania State University Applied Research Laboratory for their support. My sincerest thanks to my co-advisor, Dr. John Sustersic, for his guidance and efforts towards my thesis and research. I must thank him for continuously reviewing my thesis, and offering invaluable suggestions towards improving them.

I would like to thank my committee members, Dr. Vijay Narayanan, who is also my academic advisor, and Dr. Jack Sampson, who both have been very instrumental in my academic and professional growth and success at Penn State.

I would also like to thank my church family here in State College, Pennsylvania. I am very appreciative of Unity Church of Jesus Christ for taking me under their wing, as I strive to better myself academically, mentally, and spiritually. I also would like to thank Mrs. Patricia Hayes for allowing me opportunity to pursue my studies at Penn State and give appreciation to her, Mrs. Brenda Moore and everyone else at the Applied Research Laboratories who made my transition to Penn State much smoother.

I would like to give a special thanks to my parents, Kenny and Debra Davis, for their enduring love and support. To all of my family and friends, I love you and thank you for the encouraging phone calls and support.

Chapter 1

Introduction

Over the years, advances in computing technologies have made it possible for a plethora of computer vision applications to run on desktop systems in real time. Pan-Tilt-Zooms (PTZ) cameras are able to acquire high-resolution images and allow tracking events over a very broad range in the environment. There is an enormous amount of camera models for PTZ cameras in the market, and more are being installed each year in multiple outlets, whether these are homes, offices, or even unmanned and manned vehicles and robots. It is due to the cameras' capability to provide a much larger field of view, as compared to a static camera, that this camera is much sought after. The pan, tilt, and zoom operations of the camera allows users to even track any objects of interest in a particular scene.

1.1 Motivation

Many useful applications of tracking can include identifying objects, recognition of specific activities. One of the key components in being able to perform such activities, is to track particular objects in successive frames. The primary task of our research is to track objects on a PTZ camera and track the object as the object moves from the center of the camera's line of sight. A PTZ camera is one that has motorized control for rotating the camera in all directions in addition to being able to zoom in on an image. Many benefits can stem from the implementation of such a system. These include the benefit of providing developers with the tools necessary to develop UxVs, surveillance systems, and for even expansion or incorporation into a fully autonomous system.

1.2 Research Focus

This research details the development of a PTZ camera system for tracking the object of interest and signaling the camera accordingly to follow the object of interest. The first portion of this work will discuss various feature descriptors and which feature descriptor would be the best fit in the proposed tracking system for performance purposes. The work will then shift towards the development of algorithms which determine how far and in what direction should the camera turn to keep the object of interest in the center of its scope. This portion of the work also includes a prediction system based on a sequence of frames and the amount of object displacement that occurs in between frames.

1.3 Goals

Automatic tracking can be used in numerous applications. The goal of the work documented is to set up a real-time tracking system to be implemented on a PTZ camera. The test scenario is designed to show fulfillment of the Pan-Tilt-Zoom operations of the camera while tracking a test subject. Specifics for fulfillment are provided in further detail based on the given and some assumptions and the desire to design such a system capable of accomplishing consistent target tracking. For each test subject, the PTZ camera system results will have to be compared to truth data to meet expectations.

Chapter 2

Background

2.1 Overview

In terms of computer vision, object tracking is a very important task. Some areas that utilize object tracking include:

* Vehicle navigation: navigation unmanned vehicles with path planning and obstacle avoidance capabilities.

*Automatic Surveillance: monitoring a variety of scenes to detect activities

* Human-Computer Interaction: interactions which occur between a computer and human, such as giving gestures, or tracking eye location to use as input data for computers.

* Motion-based Detection: object identification based on shape

Object tracking can become very complex due to a number of factors including:

* Real time processing requirements (e.g. Delay from Network Cameras)

* Scene Illumination Changes

* Noise in the image

* Object occlusions, whether partial or full

* Complex object shapes or motions

Many of the existing object trackers are model-free. In other words, fragments are chosen arbitrarily and not by reference to pre-determined parts based description of the target. Model-free object tracking’s main advantage is the applicability in a variety of different scenarios without requiring any domain-specific knowledge or training. It is becoming more important that algorithms handle the complete disappearance of the object for an undefined amount of time.

There have been considerable progress in making model-free object tracking methods more robust, but the problem itself is unsolved due to challenges from occlusions, clutter, and various types of appearance changes caused by variations in illumination, camera position, or the object itself.

Models that decompose objects into multiple parts prove more robust, as local changes will only affect individual parts that are lost or in an erroneous state.

Keypoint detectors such as SIFT, FAST, BRISK, and several others estimate properties such as scale and rotation out of the image data, making keypoints ideal for parts-passed object model representation. Further works on extremely fast keypoint detectors and binary descriptors have dramatically decreased the computational burden of detecting and matching corresponding keypoints, allowing for the use of keypoints in a real-time tracking system.

There are many cases where targets must be tracked across an area larger than what is visible in a single camera’s field of view. This opens up one of two solutions:

* The target is tracked across multiple cameras.

* The target is followed with a moving PTZ camera.

The later approach is often times more desirable than the former because in cross camera tracking, the target can become lost in the blind regions between cameras, as in real world environments, it is very difficult to calibrate cameras. In addition, it can be difficult to lay out cameras in such a way that there is always overlap between the fields of view of the cameras. Following a target allows for more flexibility to adjust the camera in response to certain events so that a clear view of the target of interest is obtained.

2.2 Object Detection

2.2.1 Computer Vision Based Object Detection

Many of today's technologies incorporated Computer Vision (CV) techniques for being able to identify objects for tracking in real-time. Detection is the action or process of identifying the presence of something concealed. Object detection can be broken down into components such as features, shape, appearance, representations, and classifications.

* Detection – Detection is the action of identifying the presence of a material thing can be seen. Object detection can fall into one of multiple subcategories which include segmentation-based detection, learned-based detection, point-based detection, and motion-based detection. These categories are described as point detectors, segmentation, background modeling, and supervised classifiers in Yilmaz, et al.

* Learning/Recognition – Recognition is the action of recognizing the identification of a person, place, or thing from previous knowledge. This term can often be confused with detection as it uses many of the same object detection methods in order to "recognize" objects in an image.

* Classification – Classification is the process of classifying something according to specific shared characteristics or qualities. One example is the identification of an object through a series of successive states until a given name is reached such as:

Object -> Animate -> Animal -> Reptile -> Snake -> Python -> Dark Yellow -> Female -> Name = "Betsy".

Classification tends to be very tedious and computationally expensive in specifying every detail of a particular object of interest.

2.2.2 Detection Categories

2.2.2.1 Supervised Classifiers

Recent work in feature tracking uses learning algorithms with classifiers that recognize objects in images. Supervised classifiers are typically refer to the usage of learning algorithms that have the presence of an outcome variable to guide the learning process to train classifiers.

* Neural Networks are used to reduce computational steps required during search processes. Focus is generally on face recognition. A face must initially be detected within an image before it can be recognized.

* Adaboost, or “Adaptive Boosting”, is a machine learning meta-algorithm written by Yoav Freund and Robert Scharpire. It is considered one of the most efficient algorithms in object detection where boosting is a widely used technique [42] [43]. Adaboost makes use of a small number of weak-classifiers, which then construct cascades of strong classifiers. This combination result in high accuracy rates and computational efficiency.

* Artificial Neural Networks are families of statistical learning algorithms which were inspired by the biological neural networks, or central nervous system, and are used for function approximations that depends on large number of inputs that are typically unknown. These along with support vector machines (SVM) as well as Naïve Bayes Classifiers are different method of classification and inference methods.

2.2.2.2 Background Modeling

In changing backgrounds, establishing a model can be difficult, therefore, background modeling is used typically with static cameras. A regular sequence of frames is generally used to distinguish the background from the foreground pixels as well as to provide some robust processes for interpreting the background. Piccardi provides a very detailed look at background modeling with numerous sources such as Sequential K-D approximation, Kernel Density Estimation, Temporal Median Filters, Running Gaussian Averages, and Occurrence of Image Variations [44].

2.2.2.3 Segmentation

The main objective of segmentation is to segment objects or regions from the background. It can be defined as the task of finding groups of pixels that can go together. Some popular methods include Markov Random Field (MRF), Mean Shift (MS), and active contours. Thresholding is another simple way of segmenting images. The process of thresholding segments objects in an image by using features such as color and intensity, however, the challenge is in the selection of an optimum threshold in order to obtain specific objects of interest.

2.2.2.4 Point Detection

The information about a particular point is generally contained in a descriptive vector, or descriptor, which has a voluminous number of dimensions.   Some examples of point detectors include Harris interest point detector, Gradient Location Orientation Histogram (GLOH) [45], DAISY [46], and the Histogram of Oriented Gradients [47]. Many of these approaches use a combination of methods and processing developments. The computer vision community is continually expanding on one another's ideas to develop further improving algorithms.

2.2.3 Object Representation

In terms of tracking scenarios, objects are defined as any person or object that is of interest for analysis. These objects can be represented by their shapes and appearances. These multiple representations include

* Point – represented by a centroid or a set of points; the point representation is generally suitable for the representation of objects that occupy small regions in an image.

* Primitive Geometry – represented by an ellipse, rectangle, etc.; object motion for these representation are modeled typically by translation, affine, or homography transformation. Though primitive geometric shapes tend to be more suitable for representing simple rigid objects, they can also be used for tracking non rigid objects.

* Articulated Model – represented by body parts that are held together by joints; the primary relationship between different parts is governed by kinematic motion models. In order to represent an object that is articulated, one can model the constituent parts using either cylinders or ellipses.

* Silhouette – represented by the boundary of an object, the region inside is the silhouette; Silhouette models are useful for tracking complex non-rigid shapes.

* Skeletal Model – Object skeletons are extracted through the application of medial axis transformation on the object silhouette. This model is often utilized as a shape representation for recognizing objects. In general, there is a prominent relationship between the object representations and tracking algorithms. Object representations are chosen according to the application domain. For object tracking, which appear very small in an image, point

representation is typically appropriate. For those objects whose shapes can be approximated by rectangles or ellipses, primitive geometric shape representations are a more appropriate choice.

There are several ways that the appearance of an object can be represented. Some of these shape representations can also be combined with appearance representations for the purpose of tracking. The most common representations based on appearance include:

* Probability Densities – The probability density estimations of the object appearance are either parametric, such as Gaussian or nonparametric, such as Parzen windows or histograms. The probability densities of object appearance features, such as color and texture, can be computed from the image regions specified by shape models (the interior region of an ellipse).

* Templates – Templates are formed using simple geometric shapes or silhouettes. One major advantage is that templates carries both spatial and appearance information. Templates, on the other hand, only encode the object appearance, therefore, they are generally suitable for tracking objects whose poses do not vary considerably during the course of tracking. Self-adaption of templates during tracking is possible.

* Active Models - Active models are generated the simultaneous modeling of the object shape and appearance. The object shape is defined by a set of landmarks, which can reside on the object boundary or, alternatively, reside inside the object region. For every landmark available, an appearance vector is stored which is in the form of gradient magnitude, texture, or color. Active models require a training phase as well where both the shape and its associated appearance are learned from a set of samples using, for example, the principal component analysis.

* Multiview Models – Multiview models encode different views of an object. One approach for representing the different object views is by generating a subspace from given views. Subspace approaches, such as Independent Component Analysis and Principal Component Analysis, have been used for both appearance and shape representation.

One of the critical roles in tracking includes selecting the right features. The most desirable property of visual features is the objects' distinguishability in a feature space. Many tracking algorithms typically use a combination of these features. The features include the following:

* Edges – The boundaries of objects typically generate strong changes in image intensities. Edge detection is often used to identify these changes. Edges are less sensitive to illumination changes in comparison to color features. Algorithms that track the boundary of objects use edges as the representative features.

* Color – The apparent of an object is influenced by two factors. One is the surface reflectance properties of an object. The RGB (Red, Green, Blue) color space is typically used to represent color. The other factor is the spectral power distribution of the illuminant.

* Texture – The texture includes the measure of the intensity variation of a surface that quantifies properties such as regularity and smoothness. In comparison to color, texture requires a processing step to generate descriptors. In addition, texture features are less sensitive to illumination changes compared to color.

* Optical Flow – Optical flow is the dense field of displacement vectors defining the translation of each pixel within a specific region. Optical flow is computed using the brightness constraint, which assumes brightness constancy of the corresponding pixels in each consecutive frame.

Optical flow is commonly used as a feature in motion-based segmentation and tracking applications.

2.3 Pan-Tilt-Zoom Camera Applications

Contributions using PTZ cameras continue to increase due to improvements in real-time object detection and tracking algorithms. There have been multiple studies done that discusses tracking and maintaining objects of interest in real-time. Camera motion can undesired, for the purposes such as image stabilization requirements, or desired, for tasks such as building background mosaics [28]. In each case, the principals involves finding the transform from an image to its respective reference. These transforms are generally found between frames by matching point correspondences [55] or block matching [27]. Another technique is the use of projective textures as references [56].

Features that are non-specific are very useful in general-purpose applications. These type of features generally include motion-based features. General purpose trackers allows for a greater degree of flexibility in handling multi-purpose problems and also allows for tracking of objects of nearly any shape or size. This makes an ideal situation for tracking systems that can track a plethora of objects without having the worry with changes in illumination or appearance with certain application specific methods.

PTZ cameras have been categorized into three major research areas: calibration, 3D localization, and tracking. As processing power continues to improve, new as well as existing tools have been applied in real-time situations. These tools have also been used for tracking object within a static camera's view or tracking those objects with a moving camera.

Tracking	
Planar Structure From Motion	[26]
Background Subtraction with Mosaic	[27][28][29]
CAMShift with Kalman Filter	[30][31][32][33]
CAMShift with Window Differencing	[34]
CAMShift with back/foreground properties	[35]

Region Covariance Feature Matching	[36]
Background Compensation	[37]
Face Sequences	[19]
Upper Body and Face Detector	[38]
Complementary Features with Particle Filtering	[18][39]
Multi-Target Tracking	[49][51][52][53][54]
3D Localization	
Reactive Zoom	[26]
Mosaic	[40]
Calibration	
Bundle Adjustment	[20]
Givens Rotations and an Image Pair	[21]
Intrinsic as a function of Zoom	[23]
SIFT with Displacement Constraints	[22]
SIFT to create Mosaic Image	[24]
Geometric Restrictions (Motion Recalibration)	[25]

Table 2.1: Pan-Tilt-Zoom Camera Applications

Heuristics, techniques designed for producing solutions in a quicker fashion when classic methods are considered too slow, have often been used to design detectors for the purposed increasing speed or accuracy. Some particular applications in terms of pan, tilt, and zoom cameras include tracking the color on specific shapes [32]. Other methods are often used for detecting objects such as MeanShift, CAMShift [30] [32] [35] [38], learning an object's appearance using Principal Component Analysis and the sum of a squared difference [39], skin color segmentation [48], and GMMs [35]. Some of the overhead associated with application specific algorithms generally include increased setup time, decreased system versatility, and some feature change limitations.

Work has also been done on multi-target tracking. One work proposed the Rao-Blackwellized PF for the purpose of multi-target tracking. According to their work, the authors assume that finite sets of motion models are able to sufficiently address target tracking issues and select the Interacting Multiple Model Filter due to its performance [49]. Further work includes methods of jointly estimating object tracks, estimating the corresponding 2D and 3D temporal trajectories in the reference system of a camera as well as estimating model parameters for the purpose of estimating stable tracks that can be associated to multiple object IDs [51]. These same authors also developed a system capable of detecting and tracking people and objects from image and depth sensors on mobile robots [52]. Other methods of multi-target detection and tracking includes the use of particle filters [53] [54].

2.4 Summary

This chapter provided a brief overview of object detection and tracking, computer vision techniques and their respective Pan Tilt Zoom applications. The Pan Tilt Zoom applications and tracking forms the basis for ever developing work in building Pan Tilt Zoom computer vision systems for performing a vast assortment of tasks. Common tasks include Mean-Shift, Optical Flow and differentiating forms of background subtraction.

Chapter 3

Computer Vision Tracking Techniques

Many algorithms are available for detecting and tracking objects, and the field continues to expand in terms of speed and complexity. Many of the algorithms chosen were based on availability, popularity, and programmability. Real-world camera feeds with varying locations provide comparison data for experimentation and testing with many of the following metrics and methods. This chapter compares the different available image/video processing techniques used on stationary as well as moving cameras.

3.1 Motion Detection

Motion detection can be defined as the process of detecting changes in the position of an object relative to the object's surroundings. Motion detection can be achieved through several means including sound, infrared, magnetism, radio frequency, vibrations, and optics. The following sections will discuss motion detection as they are related to the field of computer vision.

3.1.1 Optical Flow

Optical flow is the pattern of apparent motion of surfaces, objects, and edges within a visual scene caused by the relative motion between an observer, whether a camera or an eye, and the scene.   The estimation of motion is possible through the use of sequences of ordered images. These estimations are either discrete image displacements of instantaneous image velocities.   The optical flow method attempts to calculate the motion that occurs between two image frames which are taken at times t and t+Δt at each voxel position.

3.1.2 Gaussian Mixture Model

The Gaussian Mixture Model is a parametric probability density function that is represented as the weighted sum of Gaussian component densities. These models are generally used as parametric models of the probability distribution of continuous measurements or features in a biometric system. Parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum *A Posteriori* (MAP) estimation from well-trained prior models. The model with M components is represented in the following equation:

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} w_i \, g(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i),$$

where x is the D-dimensional continuous-valued data vector(measurement or features), w_i,i=1,….,M, are the weights of the mixture, and $g(x|\mu_i,\Sigma_i)$,I = 1,….,M are the component Gaussian densities. Every component density represents a D-variate Gaussian function of the following form:

$$g(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)' \, \Sigma_i^{-1} \, (\mathbf{x}-\boldsymbol{\mu}_i)\right\},$$

with the mean vector μ_i and the covariance matrix Σ_i. The entire model is parameterized by mean vectors, covariance matricies, and mixture weights from all component densities. These parameters are then collectively represented by the following notation:

$$\lambda = \{w_i, \, \boldsymbol{\mu}_i, \, \Sigma_i\} \qquad i = 1, \ldots, M.$$

3.1.3 Point Selection and Averaging

Optical flow requires point selection. A region about the desired tracking point, *tp*, is selected initially to establish points to observe through a sequence of frames. Calculations of tracking points used for error analysis is equated by summing up the foreground point location for the Gaussian mixture model and flowpoints for optical flow in $I_{(i+1)}$ and dividing them by the overall number of foreground points or flowpoints. Those are expressed as the following:

$$tp = \frac{1}{n_{FG}} \sum_{i=1}^{n_{FG}} \mathbf{x}_{FG_i}$$

$$tp = \frac{1}{n_f} \sum_{i=1}^{n_f} \mathbf{x}_{f_i}$$

where n_f is the number of flowpoints, n_{FG} is the number of foreground pixels, x_f is the pixel coordinate of a flowpoint, and x_{FG} is the pixel coordinate of a foreground model.

3.2 Feature Detection

Feature detection is the process of obtaining image information and making local decisions at each image point whether or not there is an image feature of a specific type. There are multiple categories of image features which can fall under one of four categories.

* Ridges – Ridges are idea for elongated objects. Ridges are often defined as a one-dimensional curve representing an axis of symmetry, and has an attribute of local ridge width that is associated with every ridge point. Unlike the other feature detection categories, it is generally more difficult to extract ridge features from general classes of grey-level images.

* Corners – Corner algorithms were developed so that explicit edge detection would no longer be a requirement, for example by analyzing the curvature of an image gradient. These corners were also found being detected on parts of the image that were not corners in a traditional sense. An example of this includes a small bright spot surrounded by a dark background. These points are commonly known as interest points, but "corner" is more generally used.

* Edges – Edges are defined as the points where there is an edge or boundary that exist between two different image regions. Edges are typically are arbitrarily shaped, and can include junctions. These algorithms generally place a degree of constraints on gradient value, shape, and smoothness properties.

* Blobs – Blobs provide a unique description of image structures in terms of regions, as opposed to corners that are point-like similar to the other categories discussed above. Blobs, however, can contain a local maximum, or preferred point, meaning that blob detectors can also be regarded as interest point operators. With a shrinking image, for example, the detectors responds to points which are sharp in the shrunken image, but may be smooth in terms of the original image. Its here that the difference between a blob detector and a corner detector may become somewhat obscured. This can however be remedied through an appropriate notion of scale.

The following section will discuss the different type of feature detectors that were compatible with the applications that are discussed in this thesis.

3.2.1 SIFT

Scale-invariant feature transform (SIFT) was published by David Lowe in 1999. SIFT is able to detect features that are invariant to image scale and rotation, and has been shown to provide robust matching across a substantial range off distortion, illumination changes, noise, and changes in 3D viewpoints. Keypoints of objects are extracted from a set of reference images that are stored in a database. The object is then recognized in another image by individually comparing each feature from the new image of the database and finding the candidate matching features, which is based on the Euclidean distance of their feature vectors.

3.2.2 SURF

Speeded Up Robust Features (SURF) is partly inspired by the SIFT descriptor. The authors claim that SURF is less sensitive to noise than SIFT. SURF also requires less computational time which makes SURF an attractive feature detection algorithm. Rather than using Gaussian averaging for the image, squares are used for approximation.

$$S(x,y) = \sum_{i=0}^{i<=x} \sum_{j=0}^{j<=y} I(i,j)$$

SURF uses BLOB, based on the Hessian, due to its high accuracy, to find points of interest. Training is executed per training image through a process of visualizing each keypoint detected by SURF, manually excluding keypoints not on the object of interest.

3.2.3 FAST

Features from Accelerated Segment Test (FAST) is a corner detection method, used to extract feature points and can later be used to track and map objects. This detection method uses a circle of 16 pixels to classify whether or not a point *p* is indeed a corner. Each pixel is label from 1 to 16 clockwise. If a set of N pixels that are neighboring in the circle are brighter than the intensity of candidate pixel *p* (denoted by I_p) plus the threshold value *t* or all darker than the intensity of candidate pixel *p* minus the threshold value *t*, then *p* is classified as a corner.

3.2.4 GFTT

Good Features to Track (GFTT) is a feature selection criterion that is optimal by construction due to it being based on how the tracker works, and a feature monitor method that is able to detect occlusions, disocclusions, and features that do not correspond to points in the world. GFTT is a solid method for determining affine changes by utilizing the Newton-Raphson minimization procedure, similar to what has been done for a pure translation model [57]. This tracker shows that features with good texture properties can be defined through optimizing the tracker's accuracy.

3.2.5 BRISK

Binary Robust Invariant Scalable Keypoints (BRISK) is a 512 bit binary descriptor that is used to compute the weighted Gaussian average over a selected pattern of points near the keypoint. It then compares the values of specific pairs of Gaussian windows, leading to either a 1 or 0, depending on which window in the pair was greater. This then creates binary descriptors that work with hamming distance rather than Euclidean, and can be made to run extremely quickly using special SSSE hardware instructions for a speedup of up to 6x.

3.2.6 FLANN

Fast Library for Approximate Nearest Neighbors (FLANN) is an algorithm that is used with both SIFT and SURF. This system consist of a library of nearest neighbor algorithms from which FLANN then automatically determines the most appropriate algorithm and parameter settings given a specific database. There are a variety of settings that can be changed using FLANN. One of the initial parameters if the type of search to perform. There is a kd-tree search, a k-means search, a linear, brute-force search, and an auto-tuned search that selects one of the previously mentioned searched and tunes the parameters.

Chapter 4

CMT

4.1 OpenTLD / CMT

The capability of tracking objects is at the core of any application that is intended on understanding raw video data. Tracking-Learning-Detection (TLD) is a real-time algorithm developed for the use of tracking unknown objects in video streams. This system was developed by Zdenek Kalal during his PhD thesis at the University of Surrey. The object of interest is defined by using a bounding box in a single frame. TLD synchronously tracks the object of interest, learns its appearance and detects it whenever it appears in the video. The result is a real-time tracking system that can improve over time. The ROS implementation of OpenTLD consists of two nodes: the tracker node which use the OpenTLD library and an interface node that allows you to select a bounding box, start and stop the object tracking, start and stop the learning, import or export a model, change the tracker's method and clear the background and.

Consensus-based Matching and Tracking of Keypoints (CMT) is an updated more precise version of OpenTLD. CMT is a novel keypoint-based method for long-term model-free object tracking in a combined matching-and-tracking framework. This algorithm is able to track a wide variety of object classes in a multitude of scenes without the need of adapting the algorithm to the concrete scenario in any way. The main idea behind CMT is breaking down the object of interest into small parts, or keypoints. In order to localize the object in each frame, each keypoint casts votes for the object center. In each frame, the program finds the keypoints that were already there in the initial selection of the object of interest. This is done using two different methods. First, the program tracks keypoints from the previous frame by estimating what is known as its optic flow. Secondly, the algorithm matches keypoints globally by comparing their descriptors. Compared to other approaches, this algorithm excels when the goal is high accuracy. [8] This algorithm attempts to decouple static and dynamic elements in a radical fashion by basing the appearance model on the first frame only and by addressing changes in appearance by employing adaptive tracking techniques.

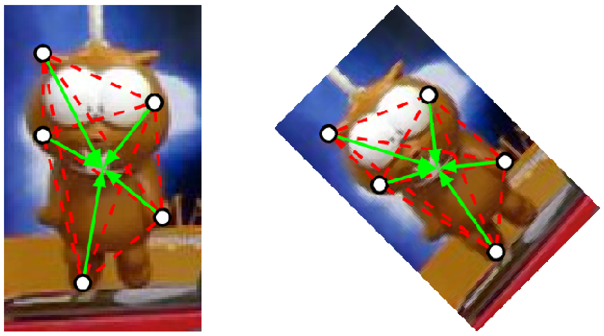


Figure 4.1: CMT Breakdown

Given a sequence of images $I_1,\ldots\ldots,I_n$, as well as an initializing region b_1 in I_1, CMT recovers the pose of the object of interest or indicates that the object is not visible.

The object model is based on a set of keypoints:

$$O = \{(r_i, f_i)\}_{i=1}^{N^O},$$

Each keypoint denotes a location in template coordinates and the descriptor f. O is initialized by detecting and describing keypoints in I_1 that are inside the initializing region b_1, then followed by a mean-normalization of the keypoint location. In order to recover the object pose, in each It with t>2, the authors were interested in finding the following set of corresponding keypoints.

$$K_t = \{(a_i, m_i)\}_{i=1}^{N^{K_t}},$$

In order to locate the object, each keypoint will cast a vote h(a,m) -> R^2 for the object center, resulting in the following set of votes:

$$V = \{h(a_i, m_i)\}_{i=1}^{N^{K'}},$$

In the simplest form, the translation changes of the object is considered only:

$$h^T(a, m) = a - r_m,$$

where r_m represents the relative position of the corresponding keypoint of *O*. When the scale of an object changes, votes have the tendency to either overshoot the object center of fall short. This limitation is overcome by scaling the votes by a single scale factor of *s*, therefore the previous equation will become:

$$h^S(a, m) = a - s \cdot r_m.$$

When the location *a* or the model index m of an entry in K' is incorrect, the votes will not target the object center, but instead point to arbitrary image locations. If V^c contains less than theta * |O| elements, it is assumed that the object is not visible. Otherwise, the votes in the consensus cluster are turned into an estimate for the object center where n = $|V^c|$.

$$\mu = \frac{1}{n} \sum_{i=1}^{n} V_i^c,$$

For the final output, the authors compute a non-axis-aligned bounded box by transforming the four corners $c_1,\ldots,c_4$ of b_1 by

$$c_i' = \mu + s \cdot Rc_i,$$

where R is the rotation matrix

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

It has been shown through an extensive evaluation that CMT achieves state-of-the-art results on a great number of sequences. The approach was compared to other "state-of-the-art" tracking approaches such as SB (Semi-supervised online Boosting [13]), TLD (Tracking-Learning-Detection [14]), STRUCK (Structured output Tracking [15]), FT (Fragments-based Tracking [16]), and LM (LearnMatch [17]).

4.2 Comparison to Other Trackers

This algorithm is compared to the other state-of-the-art tracking approaches such as STRUCK, TLD, FT, LM, HT, and SB. The algorithms were tested on 20 different sequences ranging from track running, to car chases.

Sequence	CMT	Structured -Output Tracking (STRUCK)	Tracking Learning Detection (TLD)	Fragments-based Tracking (FT)	LearnMatch (LM)	Hough-Track (HT)	Semi-supervised online Boosting (SB)
Liquor (Low)	0.91	0.74	0.70	0.46	0.86	0.07	0.43
(Medium)	0.89	0.66	0.68	0.44	0.84	0.00	0.43
(High)	0.85	0.49	0.23	0.38	0.70	0.00	0.43

Table 4.2.1: Tracker Comparison Chart (Liquor)

Sequence	CMT	Structured -Output Tracking (STRUCK)	Tracking Learning Detection (TLD)	Fragments-based Tracking (FT)	LearnMatch (LM)	Hough-Track (HT)	Semi-supervised online Boosting (SB)
Person Crossing (Low)	0.76	0.51	0.86	0.88	0.80	0.18	0.96

(Medium)	0.70	0.42	0.70	0.66	0.75	0.10	0.91
(High)	0.58	0.12	0.10	0.15	0.42	0.04	0.16

Table 4.2.2: Tracker Comparison Chart (Person Crossing)

Tables 4.2.1 and 4.2.2 show two samples of test results the authors obtained from their experiments. Their algorithm achieved the highest recall in 9 (low acc.), 11 (medium acc.) and 12 (high acc.) out of the 20 sequences and attains the highest average recall in all categories. Experimental evaluations demonstrated that this method is able to achieve state-of-the-art results and that it this algorithm is especially well suited when high accuracy is desirable.

Chapter 5

Tracking Methodologies

5.1 Tracking

When the C++ port of the CMT algorithm is running, the camera is signaled to take one shot. Once the shot has been taken and is displayed on screen, the user then has the option to select the object to be tracked. Once a target object has been identified by the operator, the first task of the tracking module is to determine where the center point of the object is relative to the camera's point of view. These coordinate values are obtained through the CMT algorithm.

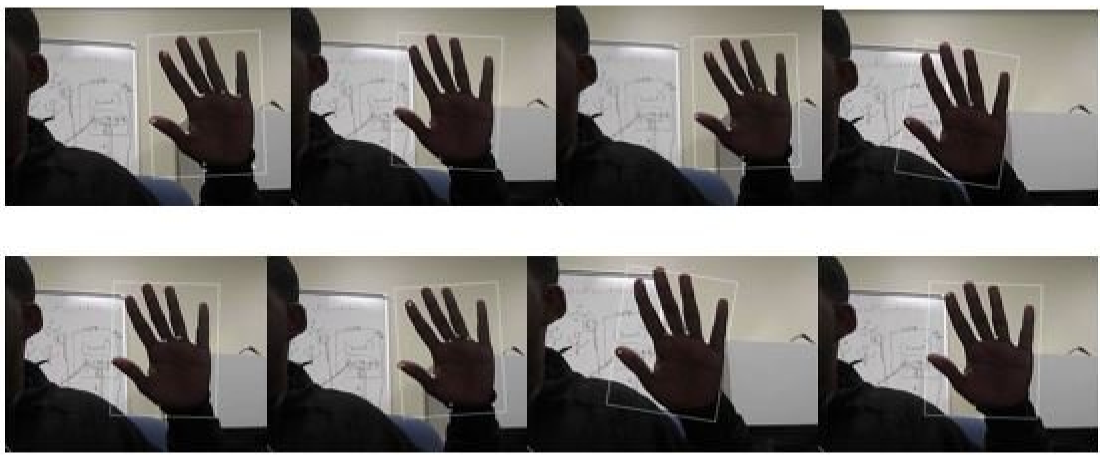


Figure 5.1: Sample of C++ CMT Port Working

Afterwards the tracker then makes a decision on whether or not the center of the object is in the center of the camera's field of view. Equations to control the pan and tilt of the camera are as follows:

$$Pan = ((CO_x - CI_x)/R)$$

$$Pan_{INV} = ((CI_x - CO_x)/R)$$

where CO_x is the x coordinate of the center of the image and CI_x is the x coordinate of the center of the object of interest. R is represented as follows:

$$R = ((CI_x - (CI_x/2))/A)$$

where A is a constant 7. This is reference point that is used to shift the camera's view to the mid-point between the center of the camera's field of view and the edge of the camera's field of view if that value were to be sent as a pan parameter to the camera. This is illustrated in Figure 5.2, where the red larger point on the horizontal axis is the midpoint between the center and the edge, and the gold point is the center of the camera's field of view. This value may vary depending on the type of camera that is being used. Similarly for the tilt control of the camera:

$$Tilt = ((CI_y - CO_y)/R)$$

$$Tilt_{INV} = ((CO_y - CI_y)/R)$$

where CO_y is the y coordinate of the center of the image and CI_y is the y coordinate of the center of the object of interest. R is represented as follows:

$$R = ((CI_y - (CI_y/2))/A)$$

where A is a constant 5. This is reference point that is used to shift the camera's view to the mid-point between the center of the camera's field of view and the edge of the camera's field of view if that value were to be sent as a tilt parameter to the camera. This is also illustrated in Figure 5.2, where the green larger point on the vertical axis is the midpoint between the center and the edge, and the gold point is the center of the camera's field of view. This value will also vary depending on the type of camera that is being used. Once the tilt and pan values have been obtained, the values are then used in signals which are sent to the camera through a Transmission Control Protocol/Internet Protocol (TCP/IP) port connection.

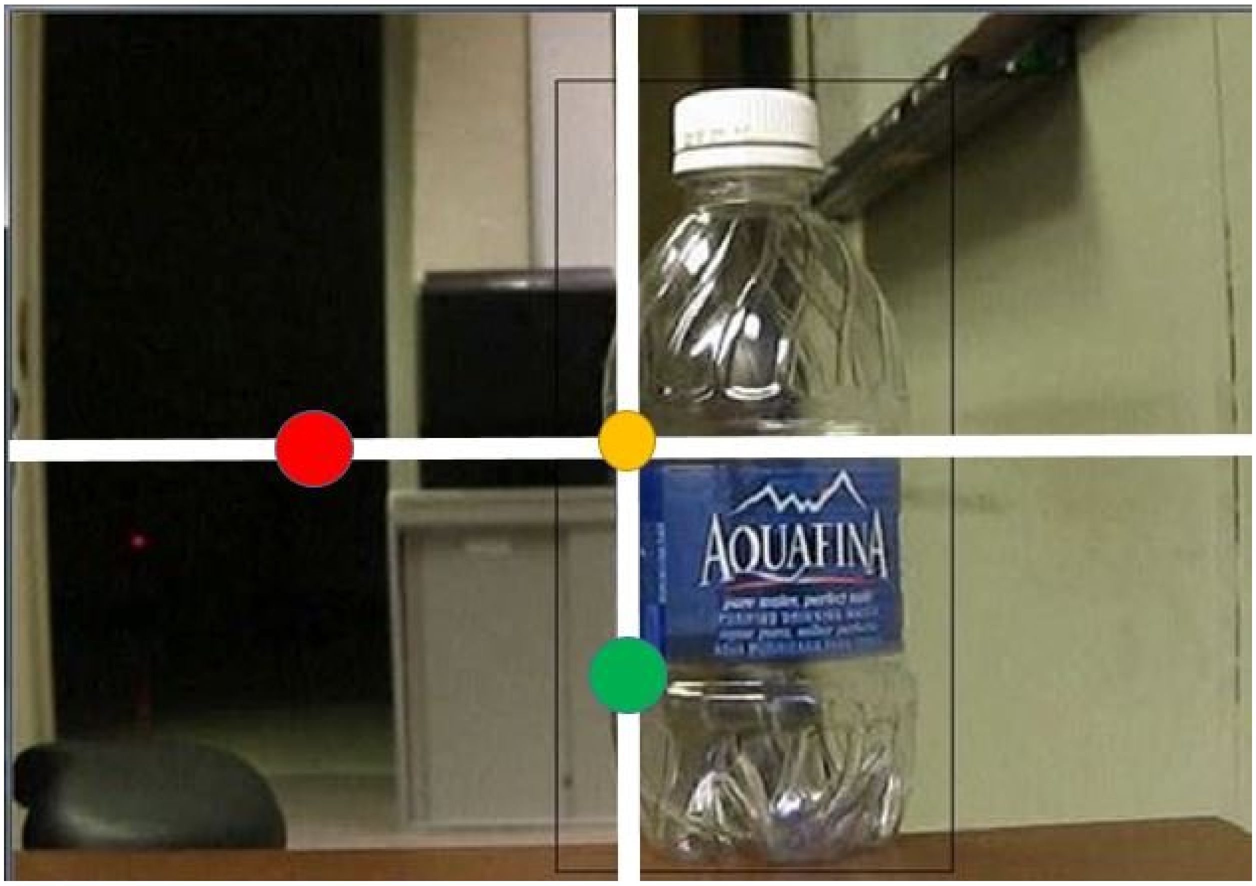


Figure 5.2: Reference Points (red = pan; green = tilt; gold = center)

5.2 Prediction

Due to the nature of the network camera, there is slightly noticeable degree of delay. Therefore, the system cannot always center the camera on the target correctly using only the determined object position. In order to compensate the motion of the target during the delays, a position predictor was used. Prediction depends on the average speed of the target over a set amount of frames, the angle difference of the object between frames, and the actual amount of delay itself. The average speed of the target for a set of frames is the following:

$$S = ((x_1 - \ldots. - x_n)/(d_1^1 - \ldots. - d_1^n))$$

where x_1……x_n are the two displacements. Motions are calculated when the camera is not moving. d_1 and d_2 represents the delay between each respective frame. Therefore, the system will put the camera center on the predicted position which is:

$$P = \beta - d_n * S$$

where β represents the original target coordinate, d_n is the average camera delay over a number of frames. P itself represents the target coordinate prediction. The camera is adaptively controlled with the speed of the network and camera response time, and the displacement of the target itself.

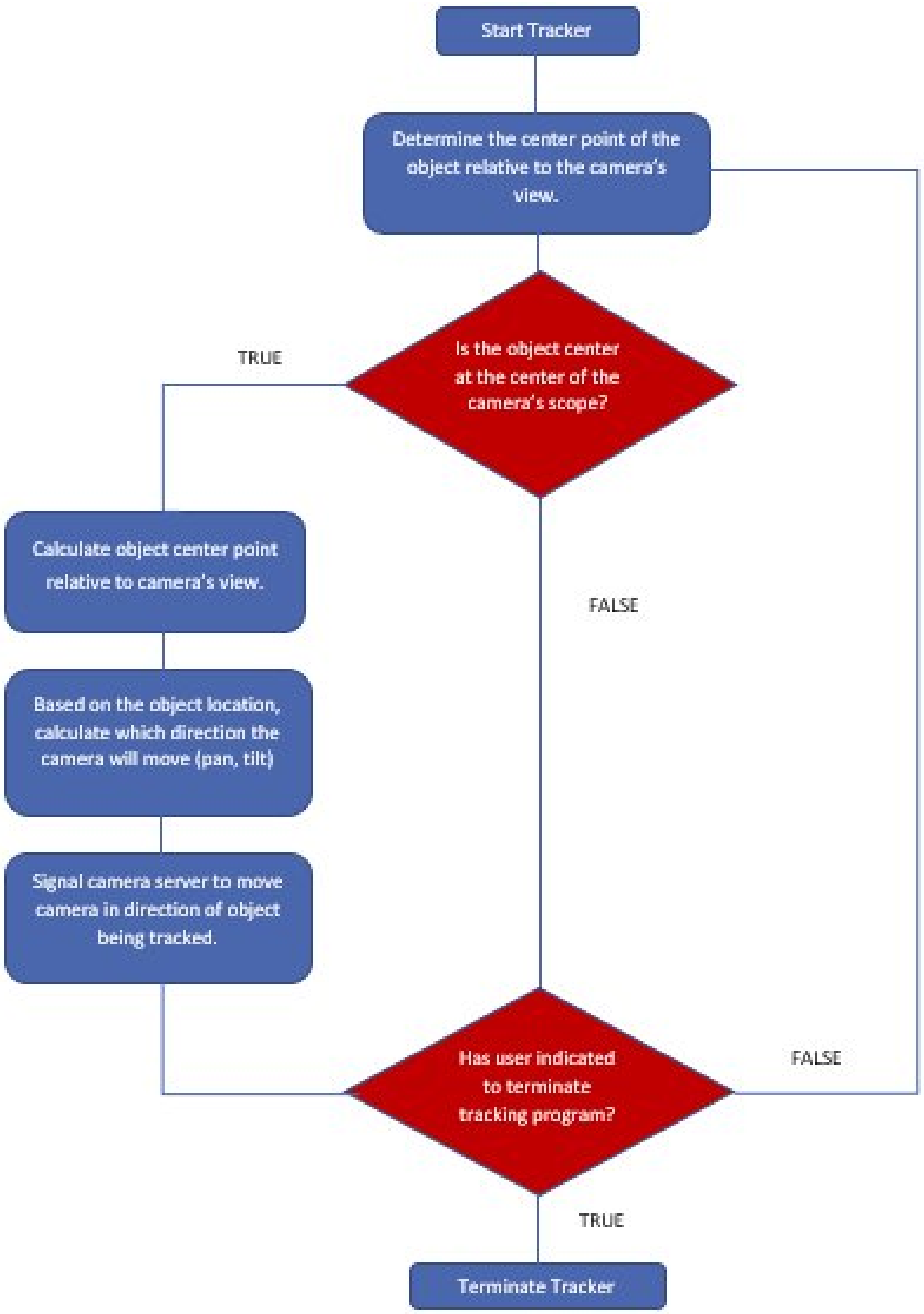

Start Tracker
Determine the center point of the object relative to the camera's view.
Is the object center at the center of the camera's scope?
TRUE
FALSE
Calculate object center point relative to camera's view.
Based on the object location, calculate which direction the camera will move (pan, tilt)
Signal camera server to move camera in direction of object being tracked.
Has user indicated to terminate tracking program?
FALSE
TRUE
Terminate Tracker

Figure 5.3: Object Tracker Flow Chart

5.3 User Interface

The interface is simple in design. Once the program has started, the PTZ camera will take an initial photo, which will be displayed on screen. From this point, the user then has the ability to select which object should be followed. After selecting the object, the user can select “t” or “T” to begin tracking that object using CMT. The object of interest will be covered with keypoints based on which keypoint detector the user decides to use as well as a box surround the object of interest. The image is broken down into four quadrants, in which the program will inform the user on which quadrant the center of the object of interest lies. The quadrants are illustrated in Figure 5.3. After each successive frame, the program will also throw out pan and tilt numbers which will be sent to the camera for the purpose of controlling the pan and tilt position of the camera. The camera continuously tracks the object of interest until the user indicates to terminate the tracker, thus closing the program. This flow is illustrated in Figure 5.2.

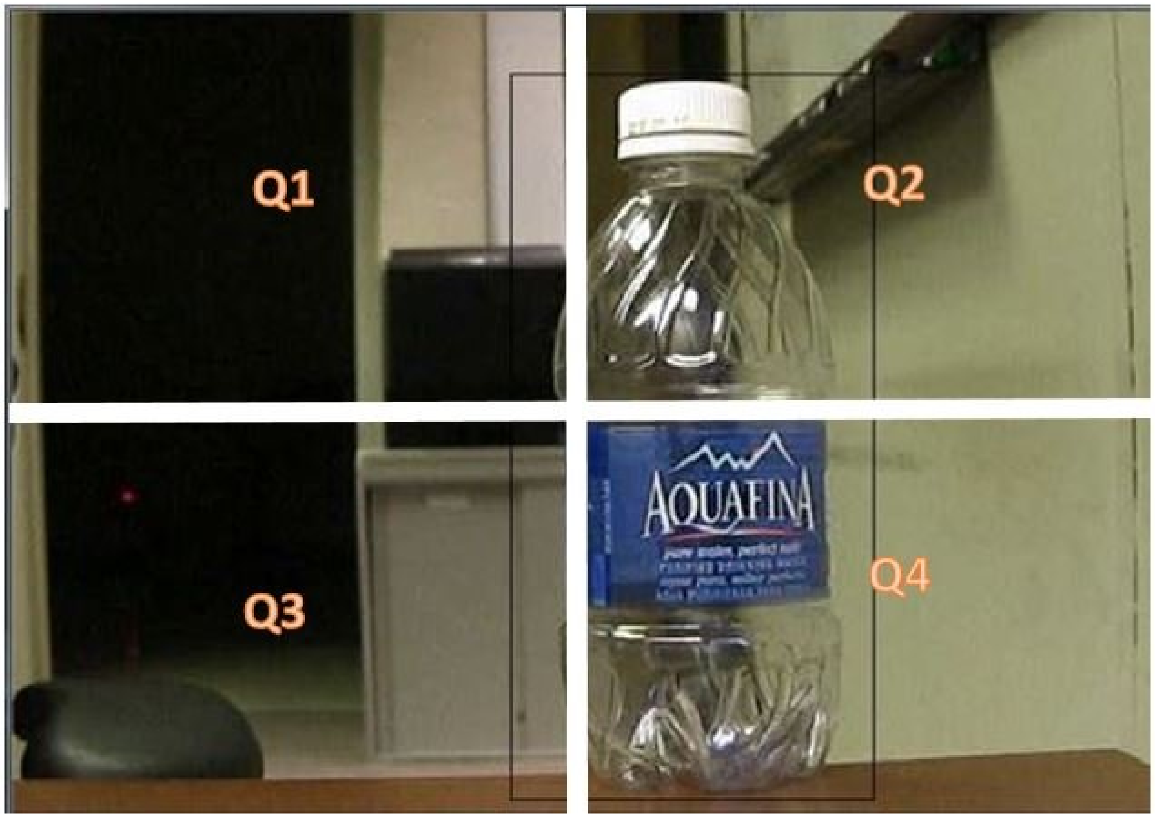


Figure 5.4: Camera Module Breakdown

Figure 5.3 illustrates the working of the camera module. The aim of the camera control module is to keep the center point of the target as close to the center of the image as possible. In every frame, the object center is calculated returned back to the user.

* If the center of the object is in quadrant I, the camera will pan right and tilt up.

* If the center of the object is in quadrant II, the camera will pan left and tilt up.

* If the center of the object is in quadrant III, the camera will pan right and tilt down.

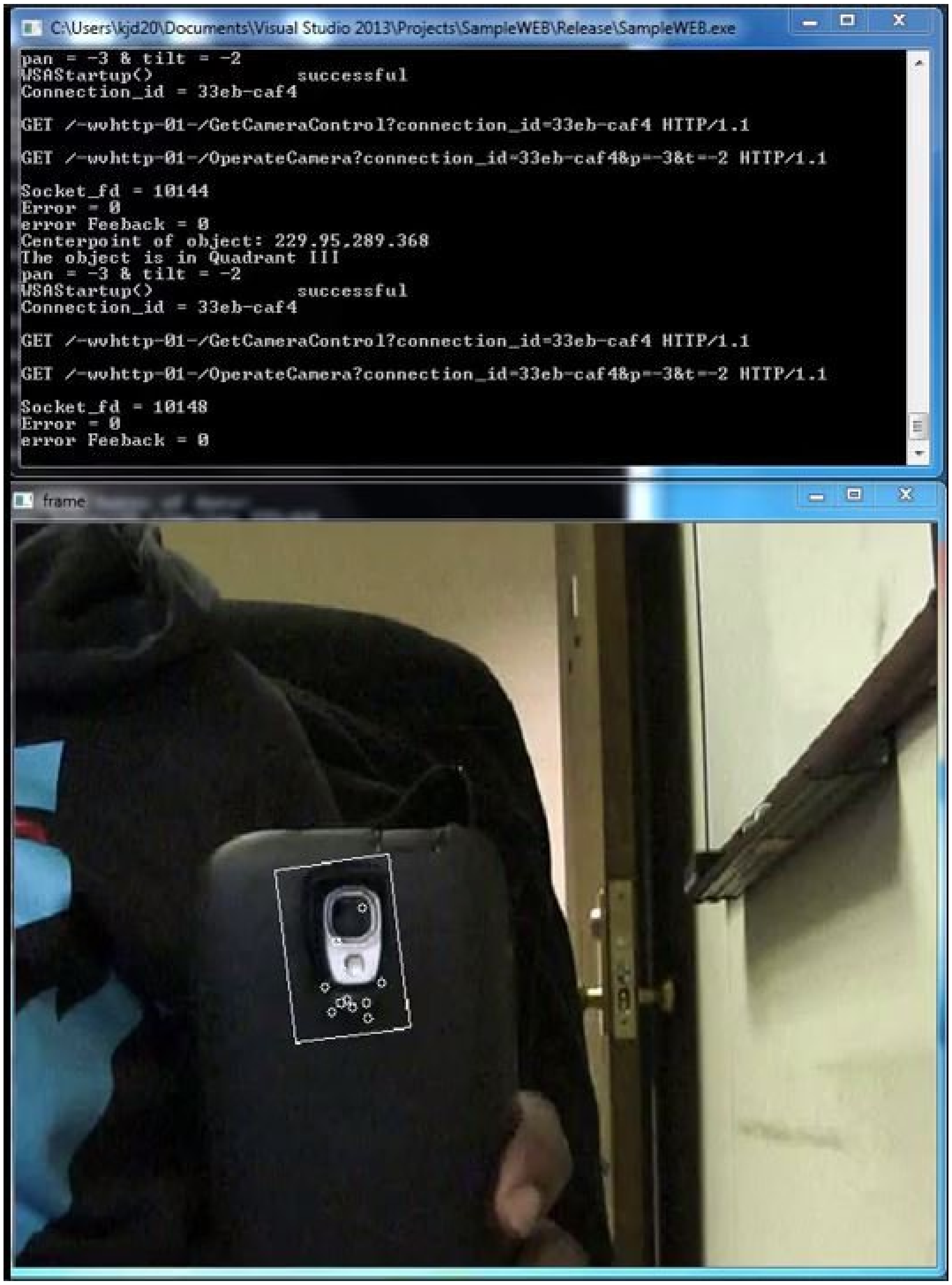
* If the center of the object is in quadrant IV, the camera will pan left and tilt down.

* If the center of the object is in between quadrants I and III, the camera will pan right.

* If the center of the object is in between quadrants II and IV, the camera will pan left.

* If the center of the object is in between quadrants I and II, the camera will tilt up.

* If the center of the object is in between quadrants III and IV, the camera will tilt down.


C:\Users\kjd20\Documents\Visual Studio 2013\Projects\SampleWEB\Release\SampleWEB.exe
frame

```
pan = -3 & tilt = -2
WSAStartup()            successful
Connection_id = 33eb-caf4

GET /-wvhttp-01-/GetCameraControl?connection_id=33eb-caf4 HTTP/1.1

GET /-wvhttp-01-/OperateCamera?connection_id=33eb-caf4&p=-3&t=-2 HTTP/1.1

Socket_fd = 10144
Error = 0
error Feeback = 0
Centerpoint of object: 229.95,289.368
The object is in Quadrant III
pan = -3 & tilt = -2
WSAStartup()            successful
Connection_id = 33eb-caf4

GET /-wvhttp-01-/GetCameraControl?connection_id=33eb-caf4 HTTP/1.1

GET /-wvhttp-01-/OperateCamera?connection_id=33eb-caf4&p=-3&t=-2 HTTP/1.1

Socket_fd = 10148
Error = 0
error Feeback = 0
```

Figure 5.5: Tracker Interface Sample

5.4 Connecting To the Camera

The camera used for experimentation utilizes WebView over HTTP protocol, or WV-HTTP. This is a protocol that provides video transmission as well as camera control function for network cameras through way of HTTP. The program initially takes a single shot and displays that on screen using GetOneShot.

* GetOneShot – GetOneShot retrieves JPEG data stream in a multi-part format. When multiple frames are specified, the maximum connection time is therefore the limit. The following URL protocol was used for GetOneShot:

http://<username>:<password>@<IP address>/-wvhttp-01-/GetOneShot?frame_count=0

For full functionality, additional signals are sent to the camera.

* OpenCameraServer(priority = ‘ ‘, video = ‘ ‘, vc_host = ‘ ‘) – OpenCameraServer creates a WV-HTTP session. When creating a privileged session, priority is specified with “priority”. This priority level is then used for access management, control privilege management, and so on. Session life span can differ depending on the priority, with privileged sessions (those with a priority level of 5 or higher) unlimited, and general sessions (those with priority levels of 0) limited to the maximum connection time, or set value.

GET /-wvhttp-01-/OpenCameraServer HTTP/1.1\r\n

* GetCameraControl – GetCameraControl requests the camera control privileges. The control privileges allocation time is determined by the session’s priority level, with privileged sessions unlimited, and others set to a finite value, or the set value.

GET /-wvhttp-01-/ GetCameraControl?connectionid=’ ‘&p = ‘ ‘& t =‘ ‘ HTTP/1.1\r\n

*OperateCamera(p = ‘ ‘, t = ‘ ‘, z = ‘ ‘, pan = ‘ ‘, tilt = ‘ ‘, zoom = ‘ ‘, focus_mode = ‘ ‘, focus_value = ‘ ‘, back_light = ‘ ‘) – OperateCamera switches and controls the camera pan, tilt, zoom, focus, and backlight correction.

GET /-wvhttp-01-/ OperateCamera?connectionid=’ ‘&p = ‘ ‘& t =‘ ‘ HTTP/1.1\r\n

* CloseCameraServer - CloserCameraServer deletes the WV-HTTP session.

GET /-wvhttp-01-/CloserCameraServer   HTTP/1.1\r\n

Chapter 6

Experiments and Application

The following section will detail the software and devices that are used for tracker experimentation. The system will be tested on multiple objects around the lab including drink bottles, mobile devices, and a set of markers that are visually similar.

6.1 Hardware

A single computer is used to control the pan, tilt, and zoom of the camera as well as the camera itself. Table 6.1 provides the computer specifications.

Figure 6.1 ASUS Ultra Etorque X1 CPU

Manufacturer	ASUS
Model	Ultra Etorque X1
Processor	Intel® Core™ i7-4770K CPU @ 3.50GHz 3.50 GHz
Installed Memory	16.0 GB
Graphics Card	Intel® HD Graphics 4600
Dedicated Graphics Memory	64 MB
Processor Cores	4

System Type	Windows 7, 64-bit Operating System

Table 6.1 Computer Specifications

The camera used is a Canon VB-C300 Network Camera. This camera is illustrated in Figure 6.1. Because of the nature of network communication, there is some delay in receiving acquired frames as well as sending control commands. Communication between the control center and the PTZ camera is executed through TCP/IP packets (i.e. all images that are captures and the control commands are sent through the network).

Figure 6.2: Canon VB-C300 Network Camera

Units	1
Manufacturer	Canon
Product Name	VB-C300
Image Sensor	¼ Type CCD (Primary Color Filter)
Max Resolution	115° (-90° - 25°)
Scanning Method	Progressive
Number of Effective Pixels	330,000 Pixels
Lens	2.4x Optical with Auto Focus (4x Digital)
Focal Length	3.0(W) – 7.2mm (T)
Pan Angle Range	340° ($\pm$170°)
Tilt Angle Range	115° (-90° - 25°)
Moving Speed	Pan: Max. 90°/sec, Tilt: Max. 70°/sec

Table 6.2: Camera Specifications

6.2 Software

The C++ compiler and programming development environment used was Microsoft Visual Studios 2013. The computer vision application programming interface used was an open BSD-licensed Software Development Kit called OpenCV version 2.4.9 [50].

6.3 Assumptions

There are several assumptions that will be made concerning the test subject as well as the environment, and several given and assumptions that are made concerning the design of the Pan Tilt Zoom camera system. The following assumptions are given regarding the test subject, PTZ camera system, and the environment:

* The test subject is unknown to the Pan Tilt Zoom system

* The environment is static during testing (i.e., no significant movements beside the test subject)

* The PTZ system contains only one camera for tracking the test subject

* The PTZ system is capable of capturing the test subject upon entrance into each camera's Field of View (FOV) after the object of interest has been selected.

* The PTZ system can track a fully or partially visible test subject

The algorithm is evaluated for tracking generic moving objects. Using the single camera without zoom, objects were tracked at approximately 1.0 m/s. In the case that the camera loses sight of the object of interest, the camera is signaled to move to the centerpoint of the last known location of the object. Once the object is back within the camera's scope, the camera will then proceed to track the object.

Chapter 7

Results

The experiments were performed using BRISK features. This decision was made due to BRISK providing the least amount latency issues as opposed to the other feature detectors mentioned earlier, while still being able to provide enough keypoints for consistent tracking. This may have resulted from some of the feature detectors being more computationally intensive. The amount of latency that occurred during tracking typically ranged between a quarter of a second to nearly three-fourths of a second. The background was generally static. The only moving objects included the objects of interest and my hand, used to move the objects. Initial experiments showed that the camera had a tendency to undershoot when following the object. This stemmed primarily from too many signals being sent to the camera during specific time frames. This promoted the use of waits to limit signals to around 3-4 commands per second. This remedied some of the initial issues. The prediction system also helped the camera keep the object closer to the center of the camera's field of view. There were some issues generated from the regular use of CMT. In the markers sequence, for example, the tracker tended to switch to another marker in the near vicinity (ex. Switching from the brown marker to the black marker and eventually back to the brown marker). When the tracker switched to another marker, the tracker would then begin following that marker. Because of this occurrence, the amount false positives were much higher in markers sequence as opposed to the phone camera and bottle sequence. This information can found in Figures 7.2, 7.4, and 7.6.

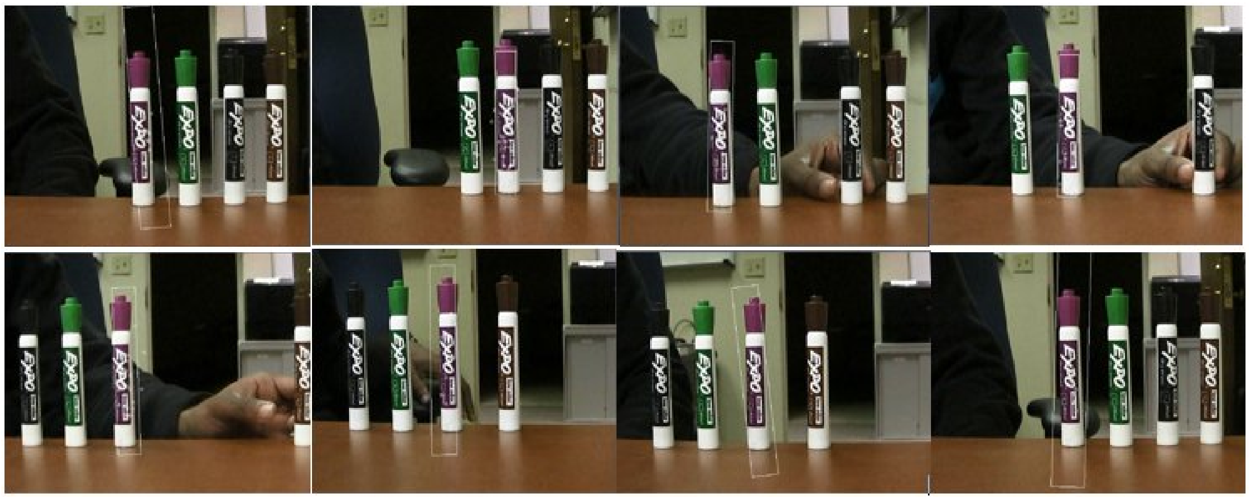


Figure 7.1: Sequence (Markers)

Other sequences include capturing other objects around the lab including plastic bottles, phone cameras, and various shapes. These are illustrated in Figures 7.3 and 7.5.

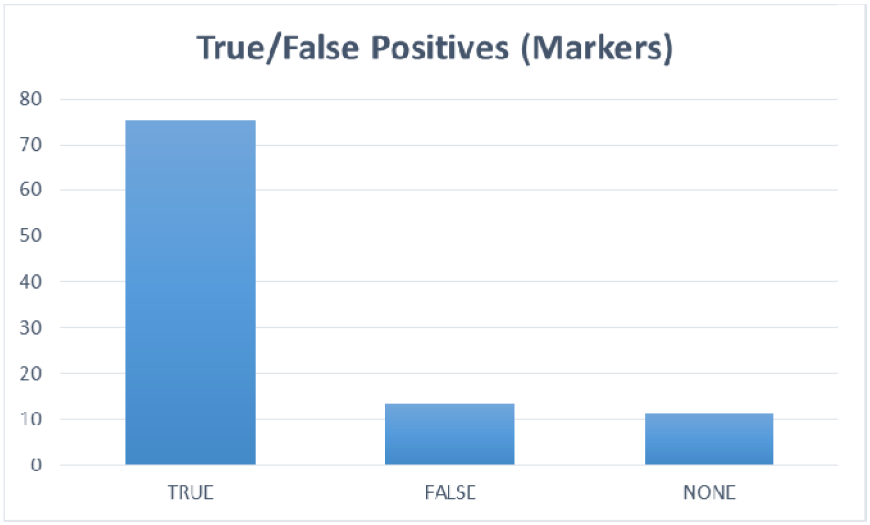


Figure 7.2: True/False Positive Chart (Markers)

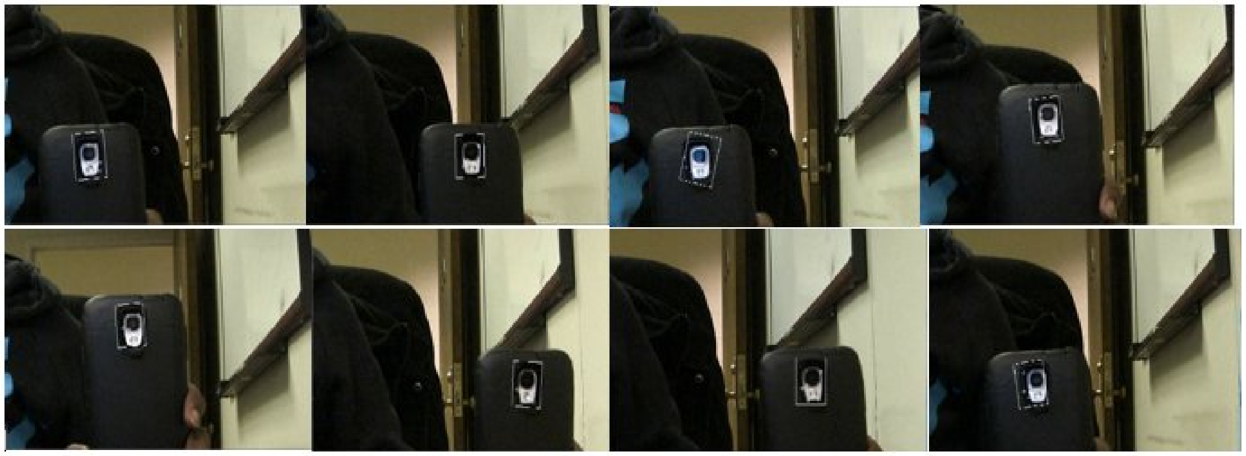


Figure 7.3: Sequence (Phone Cam)

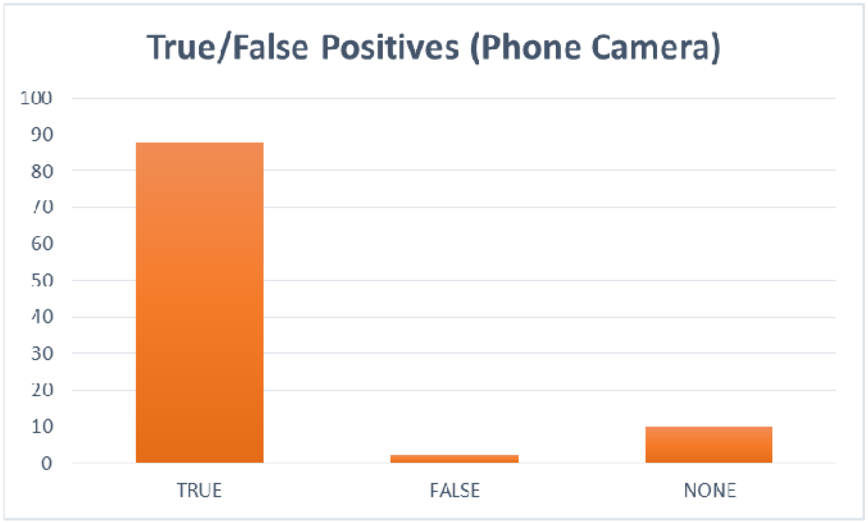


Figure 7.4: True/False Positive Chart (Phone Camera)

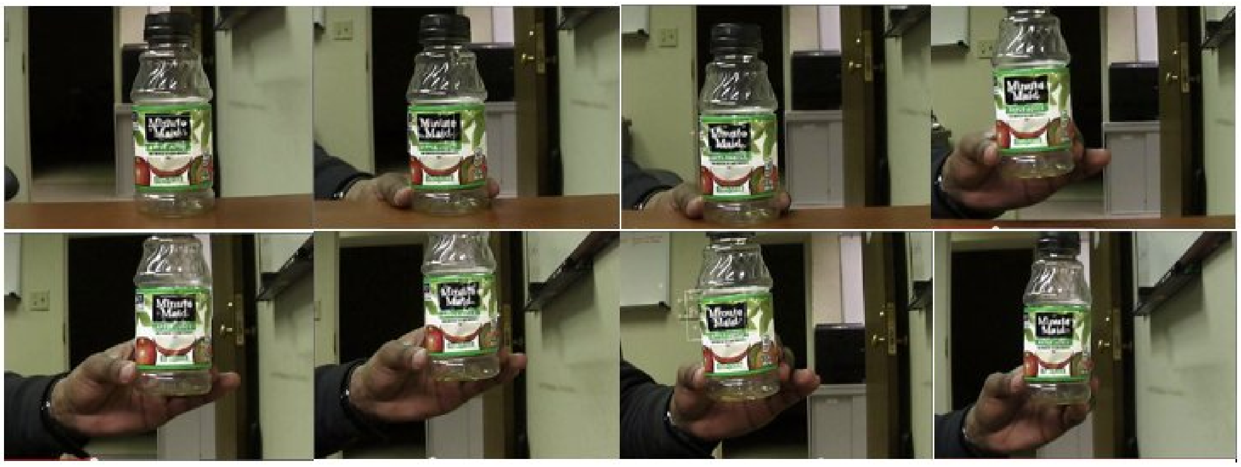


Figure 7.5: Sequence (Bottle)

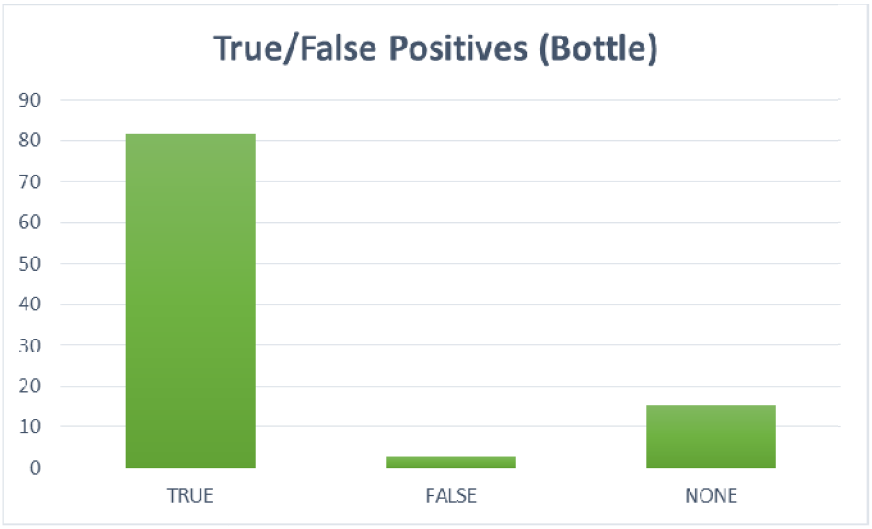


Figure 7.6: True/False Positive Chart (Bottle)

Chapter 8

Conclusions and Future Work

8.1 Conclusions

A real-time, PTZ camera system was developed. The system discussed in this work requires no training. Some of the major advantages of not having to train or set pre-defined features includes but is not limited to reducing the setup time. Other objectives completed includes the implementation of a prediction system to compensate for the delays that occurs due to the nature of the network camera. Multiple teste were conducted illustrating the ability to track a variety of objects in the environment and the ability of the PTZ camera to respond accordingly when an object of interest moves away from the center of the camera's FOV.

8.2 Limitations

Some initial lag does exist when the camera initially starts up. The lag typically last for around 5 seconds before the camera is able to consistently track the object of interest.

Due to the nature of CMT, the user is currently only able to track one item at a time as opposed to tracking multiple items simultaneously.

8.3 Future Work and Considerations

8.3.1 Design Space Exploration

Additional work can focus on the evaluation of some design alternatives for system development. Certain video sequences and feeds can often times lead to a substantial reduction in the number of tracked and active keypoints. After losing a certain percentage of keypoints, the tracker may eventually lose the entire object altogether. In this case, it would probably be feasible to consider using another object tracker that would better stabilize the overall tracking maintenance or making some changes to the CMT algorithm to prevent a box bounding the object of interest to “jump” unrealistically to another point. A substitute to that could also include tracking the object with multiple keypoint detectors for a more stable keypoint model, but with the overhead of extra computational time. Other small design choices that can be considered is the possibility of changing the camera altogether with the intention of selecting a camera that provides a smaller amount of network latency in comparison to the Canon VB-C300.

8.3.2 UxV Autonomy

We see a couple of directions we may go for future expansion of work in this thesis. Expansion of the system to function as a fully autonomous localization and navigation system for an unmanned vehicle. Using cameras on UxVs will allow for more freedom of movement. This incorporation will allow for the camera to move so that subjects remain in its Field of View, potential expanding effective capture volume as well as for better travel/flight data.

8.3.3 Depth Perception

Other considerations includes the addition of another camera for the purpose of a dual camera system. The main idea behind the camera is that additional information can be obtained from the environment such as depth. This extra depth data will then allow you to know what objects are near and far from the camera.

Bibliography

[1] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-n Learning: Bootstrapping Binary Classifiers by Structural Constraints," In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 49-56, 2010.

[2] E. Maggio and A. Cavallaro. "Video Tracking: Theory and Practice," New York: Wiley 2011.

[3] B. Babenko, M.-H. Yang, and S. Belongie. "Robust object tracking with online multiple instance learning," In *Transactions on Pattern Analysis and Machine Learning*, 33(8), 2011.

[4] H. Grabner and H. Bischof. "On-line boosting and vision," In *Computer Vision and Pattern Recognition (CVPR)*, 2006.

[5] G. Hua and Y.Wu. "Measurement integration under inconsistency for robust tracking," In *Computer Vision and Pattern Recognition*, 2006

[6] A. Adam, E. Rivlin, and I. Shimshoni. "Robust fragments-based tracking using the integral histogram," In *Computer Vision and Pattern Recognition*, 2006.

[7] L. Zhang and L. van der Maaten. "Structure preserving object tracking," In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[8] D. G. Lowe. "Distinctive image features from Scale-invariant keypoints," In *International Journal of Computer Vision*, 60(2), 2004.

[9] E. Rosten and T. Drummond. "Machine learning for high-speed corner detection," In *Proceedings of the European Conference on Computer Vision*, 2006

[10] S. Leutenegger, M. Chli, and R. Y. Siegwart. "BRISK: Binary robust invariant scalable keypoints," In *Computer Vision (ICCV), IEEE International Conference*, 2011.

[11] E. Rosten and T. Drummond. "Faster and better: A machine learning approach to corner detection," In *Transactions on Pattern Analysis and Machine Learning*, 32(1), 2010.

[12] G. Nebehay, Pflugfelder, R. "Consensus-based Matching and Tracking of Keypoints for Object Tracking," In *Winter Conference on Applications of Computer Vision*. IEEE, Mar. 2014

[13] H. Grabner, C. Leistner, and H. Bischof. "Semi-supervised On-Line boosting for robust tracking," In *Proceedings of the European Conference on Computer Vision*, 2008.

[14] Z. Kalal, K. Mikolajczyk, and J. Matas. "Tracking-Learning-detection," In *Transactions on Pattern Analysis and Machine Learning*, 34(7), 2012.

[15] S. Hare, A. Saffari, and P. H. S. Torr. "Struck: Structured output tracking with kernels," In *Computer Vision (ICCV), 2011 IEEE International Conference*, 2011.

[16] A. Adam, E. Rivlin, and I. Shimshoni. "Robust fragments-based tracking using the integral histogram," In *Computer Vision and Pattern Recognition*, 2006.

[17] S. Hare, A. Saffari, and P. H. S. Torr. “Efficient online structured output learning for keypoint-based object tracking,” In *Computer Vision and Pattern Recognition*, 2012.

[18] T. Dinh, Q. Yu, and G. Medioni. “Real time tracking using an active pan-tilt-zoom network camera,” In *IROS*, pp. 3786-3793, 2009.

[19] T. Dinh, N. Vo, and G. Medioni. “High resolution face sequences from a ptz network camera,” In *International Conference on Automatic Face and Gesture Recognition and Workshops*, pp. 531-538, 2011.

[20] I. Junejo and H. Foroosh. “Refining PTZ camera calibration,” In *Pattern Recognition, 2008. International Conference on Pattern Recognition 2008. 19th International Conference* pp. 1-4. 2008.

[21] I. Junejo, and H. Foroosh. “Practical PTZ camera calibration using Givens rotations,” *Proc. 15th IEEE Int. Conf. Image Processing ICIP 2008.* pp. 1936-1939. 2008

[22] G. Huang, Y. Tian, Y. Wang, Y. Yang, and X. Tai. “Self-Recalibration of PTZ Cameras,” *Machine Vision and Human-Machine Interface, 2010 International Conference* pp. 292-295. April 2010.

[23] Z. Wu and R. J. Radke. “Keeping a Pan-Tilt-Zoom Camera Calibrated,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35: 1994-2007, 2013.

[24] J. Zhang, Y. Wang, J. Chen, and K. Xue. “A framework of surveillance system using a PTZ camera,” *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference* pp. 658-662. 2010.

[25] B. Tamersoy and J. Aggarwal. “Exploiting Geometric Restrictions in a PTZ Camera for Finding Point-correspondences Between Configurations,” *Advanced Video and Signal Based Surveillance, 2010 Seventh IEEE International Conference* pp. 488. 29 2010-Sept. 1 2010.

[26] B. Tordoffand D. Murray. “Reactive control of zoom while fixating using perspective and affine cameras,” *Pattern Analysis and Machine Intelligence, IEEE Transactions* 26: 98-112, Jan. 2004.

[27] S. Kang, J. Paik, A. Koschan, B. R. Abidi, and M. A. Abidi. “Real-time video tracking using PTZ cameras,” *Society of Photo-Optical Instrumentation Engineers Conference Series.* pp. 103-111. April 2003.

[28] J. Zhang, Y. Wang, J. Chen, and K. Xue. “A framework of surveillance system using a PTZ camera,” *Computer Science and Information Technology, 2010 3rd IEEE International Conference* pp. 658-662. 2010.

[29] P. Azzari, L. Di Stefano, and A. Bevilacqua. “An effective real-time mosaicking algorithm apt to detect motion through background subtraction using a PTZ camera,” *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference* pp. 511-516. Sept. 2005

[30] H. Zhang, J. Hong, W. Lin, and L. Li. “Design of Tracking System Based on Meanshift and Kalman Filter,” *MIPPR 2009: Automatic Target Recognition and Image Analysis*. 2009.

[31] Z. Qigui and L. Bo. “Search on automatic target tracking based on PTZ system,” In *Image Analysis and Signal Processing, 2011 International Conference* pp. 192-195. Oct. 2011.

[32] P. Guha, D. Palai, D. Goswami, and A. Mukerjee. “DynaTracker: Target tracking in active video surveillance systems,” *Advanced Robotics, 2005. ICAR ’05. Proceedings, 12th International Conference* pp. 621-627. July 2005.

[33] USAF *Unmanned Aircraft Systems Flight Plan 2009-2047.* Technical Report, USAF, 2009.

[34] W. Yousf, O. Elmowafy, and I. Abdl-Dayem. “C18. Modified CAMShift algorithm for adaptive window tracking,” *Radio Science Conference (NRSC), 2012 29th National.* pp. 301-308. April 2012

[35] Y. Li, Q. Xie, and G. Yu. “Real-time Tracking System Combining Mixture of Gaussian Model and Continuously Adaptive Mean Shift (CAMShift) Algorithm,” *International Conference on Image Processing and Pattern Recognition in Industrial Engineering.* 2010.

[36] L. Zhang, K. Xu, S. Yu, R. Fu, and Y. Xu. “An effective approach for active tracking with a PTZ camera,” *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference* pp. 1768-1773. Dec. 2010.

[37] J. Suhr, H. G. Jung, G. Li, S. Noh, and J. Kim. “Background Compensation for Pan-Tilt-Zoom Cameras Using 1-D Feature Matching and Outlier Rejection,” In *IEEE Trans. Circuits Syst. Video Technology, Volume 21, No.3*, pp 371-377, 2011.

[38] P. Kumar, A. Dick, and T. S. Sheng. “Real Time Target Tracking with Pan Tilt Zoom Camera,” *Proc. DICTA ’09. Digital Image Computing: Techniques and Applications.* pp. 492-497. 2009.

[39] M. Akhloufi, “Pan and tilt real-time target tracking.” *In Proc. SPIE, Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications VII*, Volume 7668, 2010.

[40] K. Xue, G. Ogunmakin, Y. Liu, P. Vela, and Y. Wang. “PTZ camera-based adaptive panoramic and multi-layered background model,” In *Image Processing (ICIP), 2011 18th IEEE International Conference* pp. 2949-2952

[41] A. Yilmaz, O. Javed, and M. Shah. “Object tracking: A survey,” In *Association for Computing Machinery Comput. Surv*., 38, Dec. 2006

[42] C. Kyrkou and T. Theocharides. “A Flexible Parallel hardware Architecture for AdaBoost-Based Real-Time Object Detection,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions* 19: 1034-1047 (June 2011).

[43] S. J. Russell and P. Norvig, *Articial Intelligence: A Modern Approach, 3d Edition.* Prentice Hall, Upper Saddle River, NJ, USA, 2010.

[44] M. Piccardi, “Background subtraction techniques: a review,” *Systems, Man and Cybernetics, 2004 IEEE International Conference* pp. 3099-3104 vol. 4 2004

[45] K. Mikolajczyk and C. Schmid, “A Performance Evaluation of Local Descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*. October 2005

[46] L. Tola, and P. Fua., “DAISY: An efficient dense descriptor applied to wide baseline stereo,” In Pattern Analysis and Machine Intelligence, IEEE Transactions, 32(5):815-830, 2010.

[47] N. Dalal. and B. Triggs. “Histograms of Oriented Gradients for Human Detection,” In *Computer Vision and Pattern Recognition.* pp 886-893 2005.

[48] A. Haj., A. Bagdanov, J. Gonzalez, and F. Roca. “Reactive Object Tracking with a Single PTZ Camera,” In *International Conference on Pattern Recognition, 2010 20th International Conference* pp. 1690-1693. Aug. 2010.

[49] A. Starzacher and B. Rinner. “Embedded Realtime Feature Fusion based on ANN, SVM and NBC,” *Information Fusion (FUSION), 2009 12th International Conference* pp. 482-489. July 2009.

[50] G. Bradski, The OpenCV Library. http://opencv.org, 2000.

[51] W. Choi and S. Savarese. “Multiple target tracking in world coordinate with single, minimally calibrated camera.” *ECCV* 2010, pp. 553-567, 2010.

[52] W. Choi, C. Pantofaru, and S. Savarese. “Detecting and tracking people using an rgb-d camera via multiple detector fusion.” *In IEEE ICCV Workshops*, pp. 1076-1083, 2011.

[53] K. Okuma, A. Taleghani, N. de Freitas, J. Little and D. Lowe, "A Boosted Particle Filter: Multitarget Detection and Tracking," *Proc. European Conf. Computer Vision*, 2004.

[54] C. Hue, J. P. Le Cadre, P. P´erez “Tracking Multiple Objects with Particle Filtering” *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):791-812, 2002

[55] J. Cao, X. Xie, J. Liang, and D. Li “GPU Accelerated Target Tracking Method,” in *Advances in Multimedia, Software Engineering and Computing Vol. 1*, pp. 251-257. Springer Berlin Heidelberg, 2012.

[56] X. Clady, F. Collange, F. Jurie, and P. Martinet. “Object tracking with a pan-tilt-zoom camera: Application to car driving assistance,” *Robotics and Automation, 2001. Proceedings 2001 Internation Conference on Robotics and Automation. IEEE International Conference* pp. 1653-1658 vol.2. 2001.

[57] B. Lucas and T. Kanade. “An iterative image registration technique with an application to stereo vision,” in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, 1981.