# SeaVipers - Computer Vision and Inertial Position Reference Sensor System (CVIPRSS)

Justin Lee Erdman
*Louisiana State University and Agricultural and Mechanical College*, jchest2@lsu.edu

# *SeaVipers* - COMPUTER VISION AND INERTIAL POSITION REFERENCE SENSOR SYSTEM (CVIPRSS)

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

Division of Computer Science and Engineering

by
Justin Erdman
BS, University of West Georgia, 2010
August 2015

# Acknowledgements

This project is supported by the Army Research Office (ARO) under Grant #W911-NF1010495. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the ARO or the United States Government.

This project is also supported by the American Society for Engineering Education (ASEE) Science, Mathematics And Research for Transformation (SMART) Scholarship for Service Program. The ASEE SMART Scholarship is a part of the National Defense Education Program, an initiative of the United States Department of Defense.

I would like to thank my advisor and the rest of the department faculty and staff for all of their help and advice. I would also like to thank my friends and family for their support.

# Table of Contents

# List of Tables

# List of Figures

# Abstract

This work describes the design and development of an optical, Computer Vision (CV) based sensor for use as a Position Reference System (PRS) in Dynamic Positioning (DP). Using a combination of robotics and CV techniques, the sensor provides range and heading information to a selected reference object. The proposed optical system is superior to existing ones because it does not depend upon special reflectors nor does it require a lengthy set-up time.

This system, the Computer Vision and Inertial Position Reference Sensor System (CVIPRSS, pronounced *SeaVipers*), combines a laser rangefinder, infrared camera, and a pan–tilt unit with the robust TLD (Tracking–Learning–Detection) object tracker. In this work, a *SeaVipers* prototype is evaluated, showing promising results as viable PRS with research, commercial, and industrial applications.

# Chapter 1 Introduction

Computer Vision (CV) research has driven technological advances in important fields, such as robotics and surveillance, through developments in areas like object recognition and tracking, pattern matching, Simultaneous Localization and Mapping (SLAM), and motion planning (just to name a few), all of which are based on Image Processing and Machine Learning [1, 2, 3]. While the practical applications of such technologies are readily apparent for military and law enforcement applications—facial recognition and target tracking for surveillance; SLAM for Unmanned Aerial Vehicles (UAVs), search-and-rescue robots, etc.—recent years have shown a heightened demand for CV based technologies in commercial, industrial, and consumer products.

First debuted in 1992 by Volkswagen, self-parking and parking-assist systems are increasingly available to consumers on automobiles from many major automakers [4]. Using a combination of servo motors and many sensors, including cameras, these systems, ranging from semi-autonomous to fully autonomous, can plan and execute parking maneuvers by manipulating a vehicle's steering, throttle, and brake systems under the supervision of a small, on-board computer.

Inspired by interest within the commercial maritime community, we apply CV techniques to a product similar to parking-assist systems in automobiles, i.e. Dynamic Positioning Systems (DPS). Found primarily on large marine vessels, DPS use a variety of reference systems in conjunction with a vessel's maneuvering systems (often already automated to some extent) to manipulate propellers and thrusters to maintain position and/or heading, essentially parking a vessel on the water despite constant motion of waves and wind. Dynamic Positioning (DP) can also be used to keep position in relation to a moving object, which can be useful for pipe-laying or for vessels moving in formation. We propose an optics-based Position Reference System (PRS), called Computer Vision and Inertial Position Reference

Sensor System (CVIPRSS, pronounced *SeaVipers*), to serve as an improvement over current-generation optical PRS.

Existing positional reference sensors (Cyscan, Fanbeam, etc.) require pre-positioned reflectors as targets. These reflectors are not weatherproof; and are expensive; so they cannot be left outdoors subject to the elements for long periods of time; particularly near saltwater. So in order to maintain a fixed relative position to a target in/near the ocean, a vessel must be painstakingly moved close enough to the target to send people over with reflectors, without the aid of the dynamic positioning system. Only after this is done can the vessel safely stay near the target with the aid of the DP. A vision based system like CVIPRSS would eliminate the need for reflectors, as well as completely eliminate the initial DP-less preparation time associated with setting them up, which can be several hours.

The remainder of this work is organized as follows. Section 2.1 describes Dynamic Positioning and examines existing Position Reference Systems, Sec. 2.2 investigates Computer Vision, relevant subproblems and related works, and Sec. 2.3 examines TLD, the core CV technology for this project. In Sec. 2.4, we explore various Autonomous Vehicles, discussing development, use, and relevant works.

Chapter 3 details the problem description for developing the proposed system while Ch. 4 provides an overview of the system design. Finally, Ch. 5 discusses preliminary findings based on an experiment using an early prototype of the proposed system, which is followed by the conclusion in Ch. 6.

# Chapter 2
# Related Work

## 2.1 Dynamic Positioning

The ability to maintain a static position, otherwise known as *position-keeping* or *station-keeping*, is a universal problem among seagoing vessels. While any vessel can tie-off to the dock while in port, the question is "What to do while at sea?". Historically, marine vessels all over the world have simply used anchors, and this simple solution remains suitable for vessels today in many applications.

However, anchoring comes with drawbacks, ones that can make it particularly ill-suited for modern, advanced applications. For instance, maneuverability is severely limited once an anchor is deployed and the time to reel it back in is directly proportional to water depth. This can make the time to *anchor out* run from hours to days. Likewise, the station-keeping accuracy of anchoring diminishes in proportion to water depth. Also, the utility of anchoring is limited when the seabed is obstructed, either by natural formations or artificial features like pipelines and utility cables. Finally, it can be difficult for multiple vessels to keep position by anchoring close to each other due to safety concerns.

These drawbacks combine to make anchoring generally unsuitable for vessels attempting to dock with offshore platforms like oil-drilling rigs. To overcome these limitations, new technologies for station keeping have emerged over the past half-century, technologies collectively used for what is now called *Dynamic Positioning*.

### 2.1.1 Dynamic Positioning Systems

The American Bureau of Shipping (ABS) defines **Dynamic Positioning** as the practice of a vessel automatically maintaining position and/or heading, in a fixed location or along a predetermined track, by means of propeller/thruster force. A vessel employing DP is a Dynamically Positioned Vessel (DPV). Likewise, the ABS defines **Dynamic Positioning Systems** as "The complete installation necessary for dynamically positioning a vessel [that] comprises the following subsystems" [5]. These subsystems are:

- Power System
- Propulsion System
- DP Control System

The Power System includes all generators and or power plants responsible for the generation and distribution of electricity to the vessel, including cables and wiring. The Propulsion System includes all propellers and thrusters along with rudders, etc. which provide force for maneuvering the vessel. The DP Control System consists of all hardware and software that coordinates between other systems. For the purposes of this discussion, we divide the DP Control System into several subsystems.

- Operator Interface
- Command System
- Physical Sensors
- Position Reference Systems

The **Operator Interface System** handles input/output for the Dynamic Positioning Operator (DPO), a DPV crew member specially trained to oversee the use of a DPS. Individual reference systems can be accessed through a single, unified console, many separate, individual consoles, or a mix, depending on the installation. Direct interaction with the dynamic positioning process by the DPO is expected to be minimal.

The **Command System** (Fig. 2.1) is responsible for communication and coordination between all other subsystems. It maintains a mathematical model of the ship which (accounting for its mass, load balance, hull shape, and wind profile) describes the aerodynamic and hydrodynamic qualities of the ship. As it gathers input from **Physical Sensors**, like anemometers, compasses, or gyroscopes, the Command System will update its model with data like current orientation (roll, pitch, yaw, i.e. motion about x-, y-, and z-axes), hull draft

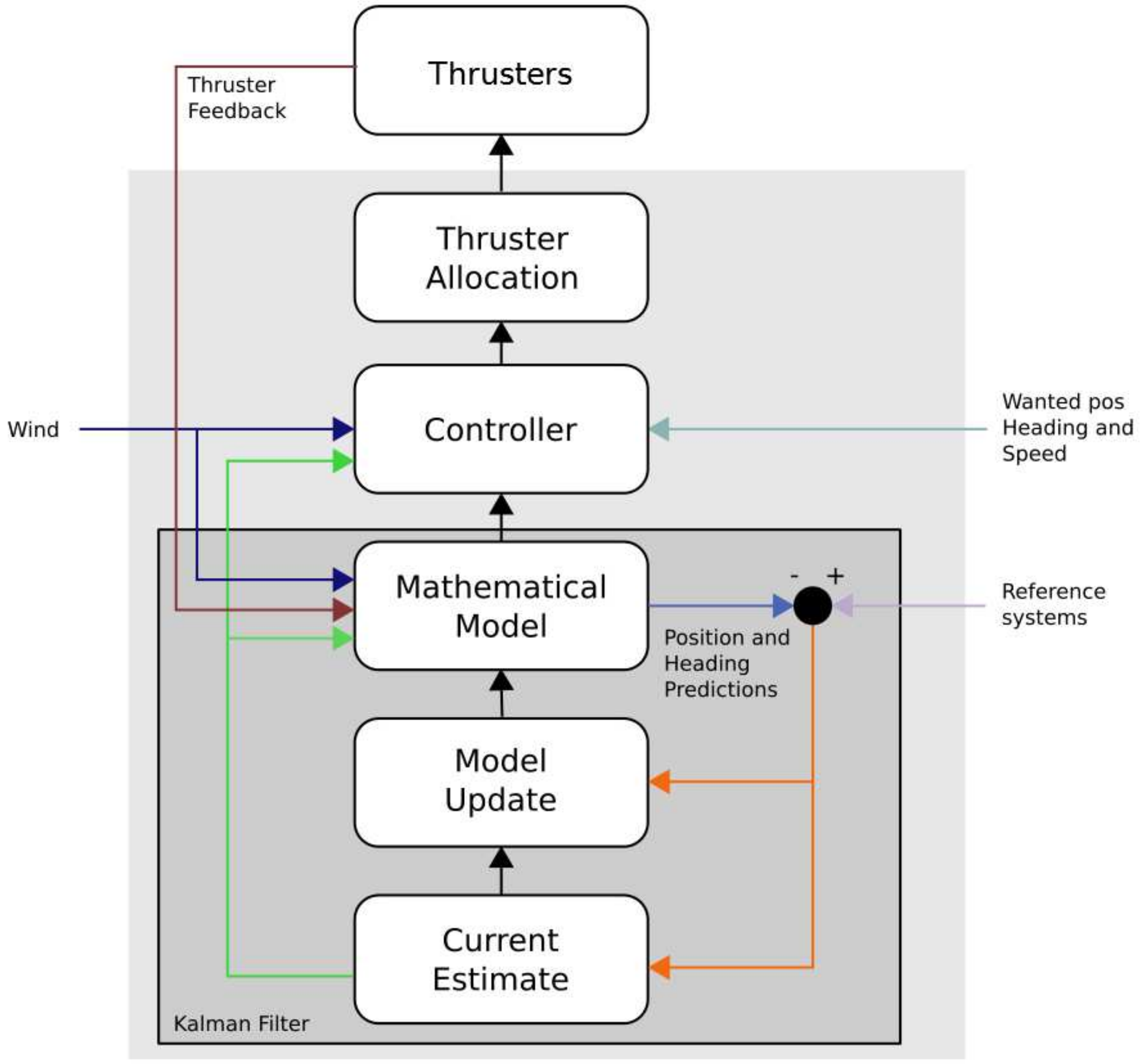


Figure 2.1: Command System

Image from: `http://en.wikipedia.org/wiki/File:Control-Kalman.svg`

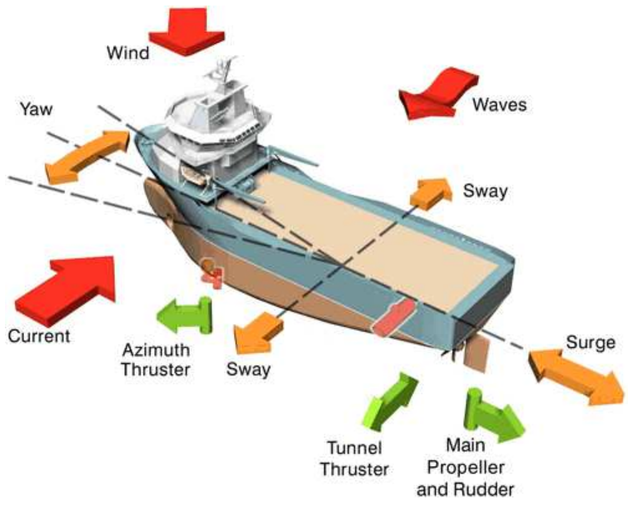


Figure 2.2: Forces acting on a marine vessel

Image from: `http://www.km.kongsberg.com`

(distance between keel and waterline), heading, or wind speed and direction. The system also tracks surge, sway, and heave. Next, after combining position estimates from each PRS, the Command System updates the vessel's current position. Finally, the Command System calculates any necessary actions to move the vessel from its current position back to the initial position, then communicates these actions with the Power System and the Propulsion System.

Fig. 2.2 shows the motion of a marine vessel and typical forces that act on it. DP is principally concerned with surge, sway, and yaw, i.e. position and orientation on the x-y plane, shown as yellow arrows. Green arrows depict forces from different thrusters (Propulsion System) and red arrows depict environmental forces.

### 2.1.2 Position Reference Systems

**Position Reference Systems** covers each PRS used on a DPV, each PRS comprised of its own subsystems and sensors. Each PRS estimates the vessel's absolute, global position by somehow determining its relative position to some object or reference location with a known position. Position Reference Systems can be categorized by sensor or reference type, such as:

- Radar
- Hydroacoustic
- Light Taut Wire (LTW)
- Optical (Cameras, Lasers)
- Global Positioning System, Differential GPS (DGPS)

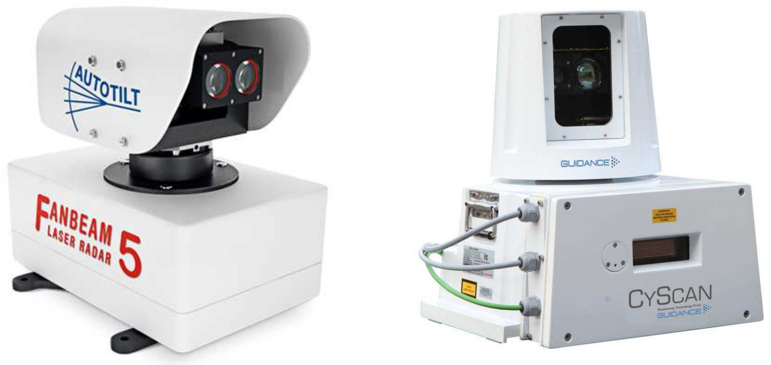
The oldest method of determining position is **dead reckoning**, which relies on a vessel's Physical Sensors. Current position is derived from the vessel's speed and course (series of heading changes), using the vessel's previous position as its reference. This method can be seriously error-prone, with errors propagating over time. Overall, dead reckoning is not sufficient for practical DP tasks, though it can supplement other PRS during poor conditions or take over as an emergency fail-safe during DPS failure.

Light Taut Wire (LTW) systems consist of a weighted sinker, a long spool of wire, a motorized winch, and a ring-shaped electromagnet. The sinker is lowered over the side of a vessel, connected by the metal wire, which passes through the electromagnet. Once the sinker reaches the bottom, the system keeps uses the winch to automatically keep the wire taut, but without lifting the sinker off of the bottom. While it does this, the system measures the angle and direction of the wire as it passes through the electromagnet and the length of wire that has been let out. This information is used to calculate the vessel's position relative to the sinker. LTW systems, though simple and effective, are limited to stationary applications and are less useful in deep water, especially in the presence of strong currents.

Radar PRS, like the Artemis Mk V or the more recent RadaScan, use microwave band radio transmissions to measure the range and heading of a prepared reference object. Artemis uses two radar antennae, one on the vessel and another mounted on the target. It also has the option of using one antennae on the vessel, with one or more active, powered beacons mounted on the target. Though it has an operating range of up to 5 km [6], Artemis antennae are heavy in comparison to other PRS and it relies on prepared targets. Antennae or beacons on the reference object, after requiring a significant installation process, must be powered, are expensive, and require regular, non-trivial maintenance. RadaScan units, while much smaller than Artemis, still require prepared targets. An array of passive (non-powered) transponders are mounted on the reference object in advance of tracking operations. Reports show that RadaScan units are limited to a range of 1 km [7], which is further limited by the angle of incidence between the unit (fixed, non-rotating mount) and the transponders. This limitation can reduce operating range from 500 m at $90\,^{\circ}$, down to as low as 50 m at $170\,^{\circ}$. Furthermore, special care must be taken when placing transponders, otherwise performance is significantly reduced. Though both radar PRS systems have the advantage of operating in all weather and lighting conditions, they both require expensive preparation of reference objects.

Hydroacoustic systems take range and angle measurements based on the known speed of sound through water, adjusting for wavelength and density. One or more transducers located on a vessel's hull broadcast acoustic signals into the water. Then, any transponders within range issue a reply. Transponders are placed on the floor of the body of water in advance at known locations. Acoustic systems are vulnerable to noise interference from any number of sources, such as the vessel's own mechanical systems and thrusters, nearby vessels and machinery, and any other acoustic systems operating in the same area.

Global Positioning Systems (GPS) would appear to make a fine candidate for PRS. Unfortunately, the resolution of standard GPS ( 15 m) is to low to provide sufficient accuracy for fine DP operations, which require sub-meter accuracy. However, there are enhancement techniques, like Differential GPS (DGPS), which can overcome some weaknesses of traditional GPS. DGPS leverages the assumption that multiple GPS receivers within close range of each other should have the same or similar error in position readings. Fixed ground-based reference stations take GPS position readings, compare them to their actual location, then compute an offset. This offset is transmitted to nearby vessels using DGPS that use the offset to correct the position reading taken from GPS. Proper DGPS can achieve an accuracy between 3 m and 5 m. However, DGPS retains many of the same limitations as GPS, like poor satellite coverage in the polar regions or disturbances in the ionosphere (e.g. sunspot activity) which is common in equatorial zones. DGPS also suffers from the *shadowing effect* caused by large metal structures which block radio and satellite signals. This is a serious problem for smaller vessels, or for any operation where a vessel has to be close to larger vessel (or platform, like an oil rig) for long periods.



(a) Fanbeam® unit (b) CyScan unit

Figure 2.3: Fanbeam® and CyScan units

(a) from: `http://www.mdl-laser.com`

(b) from: `http://marine.guidance.eu.com`

Optical PRS use light (usually infrared) to measure position. Two similar, commercially available PRS products, Fanbeam® and CyScan (Fig. 2.3), both operate by measuring the time-of-flight of infrared laser light, projected as a wide, fan-shaped beam. To take measurements, both systems rely on an array of reflectors mounted on the reference object (reflective tape, retro-reflectors, or prism reflectors) with more reflectors required at longer ranges, maximum range being approximately 2000 m.

However, existing optical systems suffer from limitations. According to the International Marine Contractors Association (IMCA), they both experience reduced operating range during heavy precipitation [8]. Fanbeam® can become confused by hits on errant reflective surfaces (reflective clothing, signs, etc.) or bright lights near the positioned reflectors and does not tolerate sunlight hitting the lens directly. CyScan mitigates these problems with improved optics and signal processing. The maximum range (2000 m) of each can only be achieved under ideal conditions using the special reflectors. CyScan, in particular, has difficulty operating beyond 400 m without use of its special prism reflectors, which are only available from the system's manufacturer, are expensive, and are also fragile. All things considered, existing laser-optical PRS are effectively limited to a range $< 500$ m for practical DP operations. Furthermore, reflectors need to be cleaned and maintained, with special care given to where and how they are mounted.

## 2.2 Computer Vision

In simple terms, **Computer Vision** is the process of applying mathematical techniques to electronic representations of visual data with the goal of deriving useful information about the contents of the visual data. Traditionally, computer vision techniques have been applied to color or monochrome versions of images captured by cameras sensitive to the range of light visible to humans, approximately 400 to 700 nm. This is generally sufficient for most common applications, though infrared light can be especially helpful in low-light situations.

With roots reaching back to the 1970s, Computer Vision is a rich and complex field with numerous existing solutions to the multitude of subproblems that combine to make up a typical computer vision based task. The remainder of this section examines relevant problems in Computer Vision and their solutions, including related works and recent developments. For additional reading or a more thorough survey, Szeliski [9] provides an excellent treatment of the topics described herein.

### 2.2.1 Features and Matching

The ability to compare images to other images (or portions of images to portions of other images) is central to many Computer Vision tasks, particularly a tracking task. The naïve approach to **Image Matching** would involve comparing the Region of Interest (ROI) of the sample image to the search image pixel by pixel at every possible location, which is obviously inefficient and slow. This approach, called **Template Matching**, is still useful in specialized cases. Several important concepts can be applied to improve the efficiency and speed of image matching.

In Computer Vision, **Image Features** are distinct regions within an image that are easy to recognize and describe, often marked by sudden/drastic changes in color or intensity. By first detecting features within an image, then limiting comparisons to only those features, the number of comparisons is *drastically* reduced. See Fig. 2.5 for an example of feature-based image matching. There are three main categories of image features: points, blobs, and lines. Each type of feature, and a given algorithms that uses it, has particular strengths. In fact, they are often complementary and higher level algorithms can use a variety of features together for a more comprehensive understanding of an image. See Fig. 2.4 for examples of each feature type.

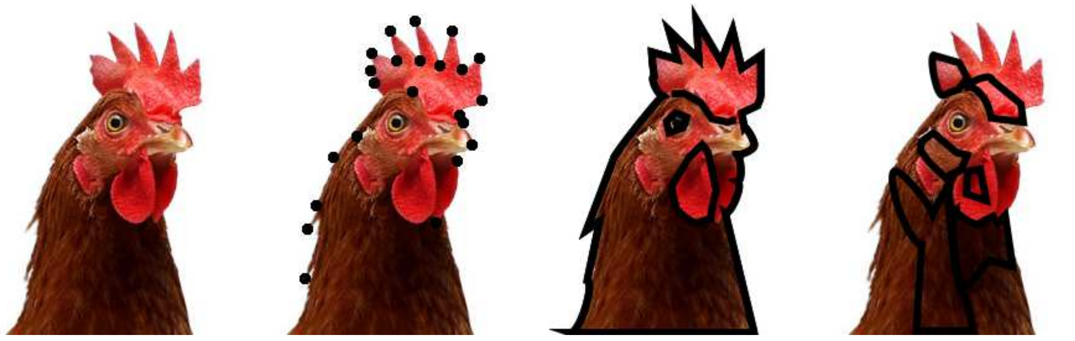

(a) Original Image (b) Corner Features (c) Edge Features (d) Blob Features

Figure 2.4: Example of Different Types of Image Features on a Chicken

Point-style features (Fig. 2.4b), often called keypoints, interest points, or corners, depending on the application, are the simplest type of features and the most widely used for *image matching*. Line-style features (Fig. 2.4c) include edges, curves, and certain geometric shapes like lines and circles. Algorithms using these features are often part of an *image segmentation* task, i.e. dividing an image into distinct regions representing discrete objects (Sec. 2.2.2). Blob-style features (Fig. 2.4d) are also referred to as regions. These features represent a sort of hybrid between line and point features, corresponding to parts of an image that neither a keypoint nor curve would fit. The algorithms used for *SeaVipers* primarily use point-style features.

For any given type of image feature, there can be many different ways to find that type of feature within and image. An algorithm that locates image features within and image is called a **Feature Detector** or **Feature Extractor**. Furthermore, there are different ways of representing features in memory. Specialized data structures, called **Feature Descriptors**, each contain the mathematical description of a particular image feature. **Feature Matchers** are the algorithms that compare descriptors from different images, looking for similarities. Just like image features, feature detectors, and feature descriptors, multiple *feature matchers* can exist to work with a given type of descriptor, and many matchers can work with several different types of descriptors, depending on the application in question.

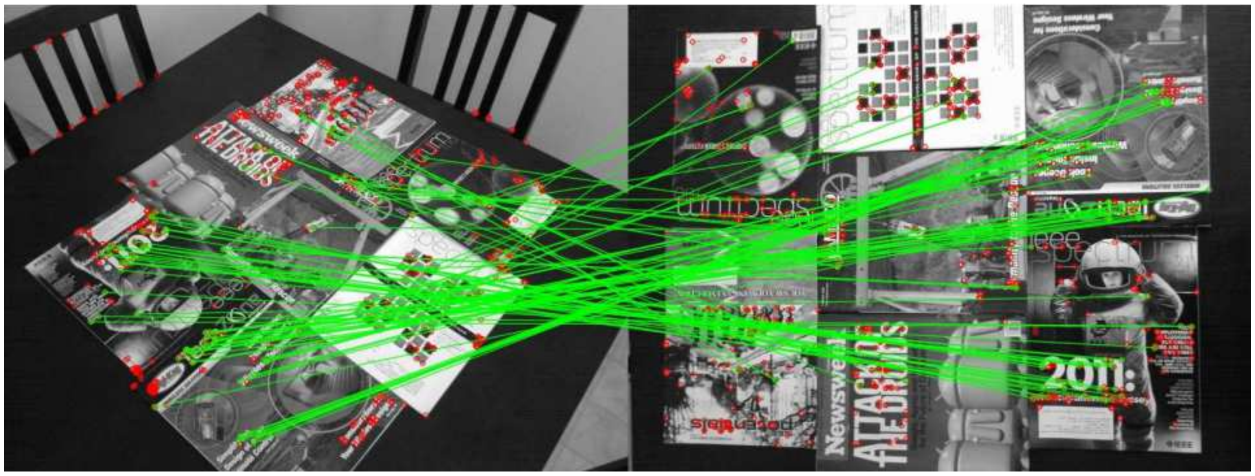


Figure 2.5: Image Matching under affine transformation using ORB

Example of an image matching using ORB. Note that image is rotated, scaled, and tilted. Green lines are drawn between matching keypoints while unmatched keypoints are red and have no connecting lines. Image from [10].

All of these concepts are used in SIFT (Scale Invariant Feature Transform) [11], an early and well-known feature-based image matching solution. SIFT detects keypoints within an image and selects *patches*, fixed-sized groupings of pixels around a given point. These patches are used as features. The most important innovation from SIFT is the way it leverages *feature descriptors* to overcome the problem of affine transformations. The descriptors used in SIFT are scale-invariant, meaning that a large feature from an object in the foreground of one image can match to a smaller feature from another image, possibly from the same object pictured farther away. SIFT descriptors are also invariant under rotation and image processing techniques can be used to overcome minor affine distortions.

SIFT has inspired numerous improvements and alternatives, like SURF (Speeded Up Robust Features) [12], which combines a faster and more efficient detector, descriptor, and matcher. SURF has the same scale-invariant and rotation-invariant properties of SIFT, but with some implementations performing image matching in half the time. It combines a *Fast-Hessian* Detector (using the determinant of a Hessian matrix) with a novel SURF descriptor (using Haar-wavelets).

The BRIEF method for matching (Binary Robust Independent Elementary Features) [13] builds upon SURF, replacing SURF descriptors with binary strings (called BRIEFs), which are smaller in memory and compared with each other using Hamming distance instead of $L_2$ norm. The Hamming distance comparison is more efficient. Image matching with BRIEF is shown to be more accurate and faster than matching done with SURF or SIFT [13]. However, BRIEF descriptors lack rotation-invariance.

The FAST (Features from Accelerated Segment Test) approach for *feature description* is an improved method of keypoint detection that drastically outperforms previous methods [14]. The improvement in speed is significant because this made FAST the first detector capable of processing video in real-time (approximately 50 Hz). FAST leverages the *Accelerated Segment Test* to quickly generate corner-based keypoints. However, FAST is weak to scale variations and does not have a measure of feature orientation like features in SIFT or SURF.

Most recently, ORB (Oriented FAST and Rotated BRIEF) was developed, building upon prior innovations [10]. ORB improves the feature detector from FAST by using pyramids to account for variations in feature scale and the Harris corner measure to reject edge-based features, which are “less interesting” than corner-based ones. ORB also includes a measure of corner orientation using *intensity centroids.* A machine learning based method is used in ORB to *decorrelate* BRIEF descriptors, which approximates the rotation-invariance property of previous descriptors. Using the new oFAST detector and rBRIEF descriptor, ORB performs faster than SURF by a factor of 10 and faster than SIFT by a factor of 100 with similar matching performance to both while being drastically more efficient and less affected by image noise. Refer back to Fig. 2.5 for an example of image matching using ORB.

### 2.2.2 Image Segmentation

**Image Segmentation** is the process of dividing an image into one or more sets of pixels, with each set containing pixels with similar characteristics, i.e. pixels that *go together* [9]. This is an old and widely studied problem in Computer Vision. As discussed in Sec. 2.2.1, edge image features and curve image features (also sometimes called contours) are useful in image segmentation because they can describe the boundaries between neighboring segments. Likewise, blob features can describe the segments themselves. However, features represent just one set of segmentation approaches.

The most simple approach is **Thresholding**, wherein each pixel is put into one of two groups (usually *black* and *white*) based on whether its value for a certain property (red, alpha, intensity, etc.) is above or below a certain *threshold* value. Though apparently primitive, this method can yield valuable results when applied intelligently, e.g. in layers or when done several times on a given image for different properties. More sophisticated forms of Thresholding exist which correct for lighting gradients and image noise.

**Clustering** techniques are also popular for segmentation. Whether agglomerative (region merging) or divisive (region splitting), these techniques are some of the oldest. Early methods use only local information when clustering, sometimes leading to unexpected results, such as when a histogram based method is caught in a local minimum or maximum. More advanced clustering techniques, called *Split and Merge* techniques, can combine and separate regions as necessary, leading to more accurate results.

The **K-means** method takes a different approach to clustering [9]. Instead of splitting or merging, a number of pixels, $k$, are chosen as segment centers either heuristically or randomly. Next, each and every pixel is grouped with its *nearest* segment center based on some distance function, usually based on color and/or intensity. After all pixels have been grouped into one of $k$ segments, the segment centers are reevaluated so that the new center minimizes the distance function computed with every other pixel in its group. The process repeats until it converges.

The **Mean-Shift** method for cluster analysis [15] is of particular interest. It implicitly models the probability density function, similar to the K-means method, but instead uses a smooth, continuous non-parametric model, efficiently finding peaks in high-dimensional data. Mean-shift is used as the basis for some simple Object Tracing algorithms (Sec. 2.2.4).

### 2.2.3 Objects: Detection and Recognition

**Objects** hold a special place in the realm of Computer Vision. Since the general goal of Computer Vision is to derive understanding of the physical, 3D world from digital, 2D images, and given that the physical world is commonly understood to contain discrete 3D objects, special care must be taken in how objects are defined and in how they are handled in a given CV application. For Computer Vision in general, *what an object is* matters less than *how an object is represented*. Initially, an object will be represented as a region within an image containing only (or almost only) pixels with visual data about that object. Often, an object is simplified to a collection of distinct features through feature extraction. This collection of representational features can then be stored in memory.

- **Object Detection**

**Object Detection** is the process of finding a particular object or set of objects within an image. This is done by using *image matching* techniques (Sec. 2.2.1) to match against a stored collection of objects. Detection is an important part of many technologies, like industrial quality control or automated surveillance. Naïve object detection would first identify features within the query image, then compare each possible object in the search database with the query image, going feature by feature. This process can be sped up using an intelligent choice of data structures to store candidate objects and by using efficient search algorithms that eliminate unnecessary feature descriptor comparisons during the image matching step. Additional speed up can come from restricting the number of features used or by restricting the search area to a particular region or regions. However, increases in speed can come at the cost of accuracy, and increases in generality (reduced false negatives) come with a decrease in accuracy (increased false positives).

Restricting the search area to a specific subregion of an image is an intuitive choice and can be highly effective given sufficient domain knowledge. For example, if all candidate objects in a search database are known to be light in color, darker regions of a query image can be automatically ignored. Consider also a street/traffic camera at an intersection that, instead of searching each entire frame for automobiles and pedestrians, only searches subregions of frames where it has detected motion (Sec. 2.2.4).

- **Object Recognition**

According to Szeliski [9],"Of all the visual tasks we might ask a computer to perform, analyzing a scene and recognizing all of the constituent objects remains the most challenging." This due to the numerous possible variations in pose, non-rigid transforms, lighting, occlusions, color, pattern, contrast, background, etc. that can affect the appearance of objects within a scene. Typically, domain knowledge about the environment and possible objects within it can help simplify recognition, but it remains daunting.

**Object Recognition** is the process of finding *and identifying* objects within an image. The recognition task occurs at different degrees of complexity, which Szeliski divides into three levels:

1. Object Detection

2. Instance Recognition

3. Object Classification

The first level, *Object Detection*, is the simplest. Images are searched for *specific, individual* objects that were encountered before and saved in a database, as described previously. An object, once matched against the database, is identified as the same object that was previously detected.

The second level, **Instance Recognition**, is more complex than the previous level. Given a particular class, type, or category of object, images are searched for *instances* of that object class. Class definitions vary by application and method, but they often rely on an **Object Model**, which is a set of characteristic features (those typical of a given class) arranged in a valid formation derived from the geometry and transformations typical to that same class. Domain knowledge is important for selecting the right features for a model. **Machine Learning** techniques often assist in model creation, taking in a large training set of images containing valid instances of a class, and, by comparing them, learning what features are most important for describing the class and how those features are typically arranged.

The third level, **Object Classification**, is the most complex and relies most heavily on machine learning. Classification is actually the reverse problem of instance recognition. Given an image containing some instance(s) of some class(es) of object(s), the object must be isolated and compared against various class definitions, then finally identified as belonging to some class (usually just one class, but possibly more). One can see that classification is comprised of multiple instance recognition tasks, with corresponding learning task for each class that needs to be recognized. Instead of matching against a database of instances, like an object detector, an object classifier would match against a database of models.

- **Practical Application**

To better understand each level and the distinctions between them, consider the practical example of **Human Facial Recognition**. In the simplest form (level one), an Object Detector considers images (like security photos or mugshots) that may or may not contain human faces. The detector has a database of human faces, e.g. known criminals for a law-enforcement application or past customers for commercial and marketing applications. When a *specific* face is positively identified, the recognition system can access database records connected to the person associated with the detected face, going on to record information and/or prompt another system to take actions, like moving a video game character on-screen.

Next, consider level two, Instance Recognition. Suppose a surveillance system tries to automatically identify human faces in a cluttered environment with different types of moving objects. After performing background subtraction and focusing only on new objects that move into the scene, the system speeds up the search process by only considering new objects that do actually contain human faces. It does this by performing Instance Recognition, extracting only *image features* typical to human faces (facial features, like nose, mouth, etc.) and considering human facial geometry, like distance between the eyes or shape of the jaw-line. Done quickly, eliminating non-face objects enhances the process by eliminating costly, useless searches by an object detector over regions that do not actually contain human faces.

Finally, consider level three, Object Classification. Continuing with facial recognition, consider an advertisement system in a department store that must choose appropriate sales and offers to display on a screen which is seen by customers as they first enter the store. Known customers are easily dealt with using recognition techniques from the previous levels. For example, frequent customers may have a photo-ID card with associated membership to the store's discount club. These customers' purchases are logged, so whenever a member's face is detected, offers related their past purchases are displayed. Perhaps the system displays athletic clothing or exercise gear to someone who previously bought running shoes. Making these choices based on purchase history is a completely separate Machine Learning task that goes beyond this discussion. But how does the system deal with new faces? It could simply default to pre-selected and most popular ads, or it could attempt make a more informed decision by *classifying* the new person based on facial characteristics. If the system had previously learned that people with long faces are more likely to socks than people with wide faces, it could attempt a binary-classification task. The system would label unrecognized faces as *Long* (likely to buy socks) or *Wide* (unlikely to buy socks), and then display an advertisement for a discount on 12-packs of black socks to man with a long face that the system had never encountered before.

For further examples, refer to [16] for a discussion of Optical Character Recognition.

### 2.2.4 Video and Motion: Object Tracking

Object Detection and Recognition seem like challenging enough tasks without considering the added difficulty of moving objects. However, while overall more challenging, deriving information from moving objects and/or moving cameras provides additional opportunities to learn about the structure and layout of objects within a scene. Furthermore, gathering data from video, as opposed to an arbitrary collection of images focused on the same scene, can actually simplify certain tasks. This is due to the *spatiotemporal* correlation between subsequent video frames. Given a known, constant frame rate (e.g. 50 or 60 Hz), a Computer Vision system can confidently assume that there is a constant interval of time between when each frame was captured (temporal correlation). A system can also assume that every frame was captured within a short distance $d$ of the last one ($d = 0$ for stationary cameras) and that any objects in the frame will be within a short distance of their previous location (spatial correlation).

The correlation assumption allows the use of **Object Tracking** instead of object detection. Since subsequent appearances of a given object are assumed to be nearby previous ones, matching is constrained to the region or *neighborhood* immediately surrounding a previous known location. After a few initial frames, an object tracker can estimate the *object trajectory*, then use that to search where a tracked object should appear next, instead of where it last was.

The Mean-shift algorithm [15] can be repurposed to track a moving object across multiple video frames. Given a bounding rectangle that defines the initial view of the target object, the *start window*, along with the initial image frame, the algorithm computes a histogram of the pixels within the window, then *shifts* the window center to the location of the *mean* value for the histogram, iterating until the window is over the target's centroid for a given frame. For subsequent frames, the algorithm passes the end location of the window from the last frame as the new starting location. If the target object has not moved, then the window will remain in-place. Otherwise, the algorithm will begin the shifting process anew, until the object's centroid is found again. In this way, the rectangular window follows the tracked object's centroid as the object moves.

However, an approach like Mean-shift is limited in several ways. First, it has no way of handling scale changes or in-plane rotations. Second, it cannot handle occlusions, out-of-plane rotations, or other transformations. The CAMshift approach (Continuous Adaptive Meanshift) [17] addresses the first set of limitations, building upon Mean-shift by replacing the static rectangle window with a best-fit rotated rectangle, which is re-sized after every frame. Orientation is found using a best-fit ellipse.

**Optical Flow** is a concept from perceptual psychology that refers to the perceived change in position of observed objects due to motion between those objects and the observer. Applied to Computer Vision, this concept is useful for motion tracking and stereo matching. Optic Flow based trackers compute flow vectors at some number of points, usually a set of point features from a feature extractor, then use these vectors to compute the overall motion of objects within an image relative to their positions in a previous image.

Several methods for computing optic flow exist, like the Horn–Schunk method [18] or the popular Lucas–Kanade method, often called L–K Optical Flow [3, 19]. The LK method assumes that flow is constant within a certain distance from any given pixel, thereby simplifying the process and solving by method of least squares. However, if the motion between two frames increases past a certain point, the flow equations cannot be solved. To overcome this, so-called *pyramidal* implementations estimate the solution by applying the method to a series of progressively shrunken copies of an original frame. These smaller, reduced quality images are thought of as stacked atop each other, forming a rectangular *step pyramid*, hence the name pyramidal L–K Optical Flow. However, optical flow methods rely heavily on clearly distinguishable backgrounds, i.e. ones with texture, patterns, or other distinct features.

## 2.3 Tracking–Learning–Detection

For any one difficult problem in Computer Vision, there exist multiple approaches to solving that problem. Furthermore, a set of solutions to a given problem tends to have members with complementary strengths and accompanying weaknesses. This is no different for Object Tracking, which has two general families of solutions: **Motion Tracking** (e.g. Optic Flow, CAMshift) and **Track-by-Detection** (i.e. "tracking" by applying detection at each frame).

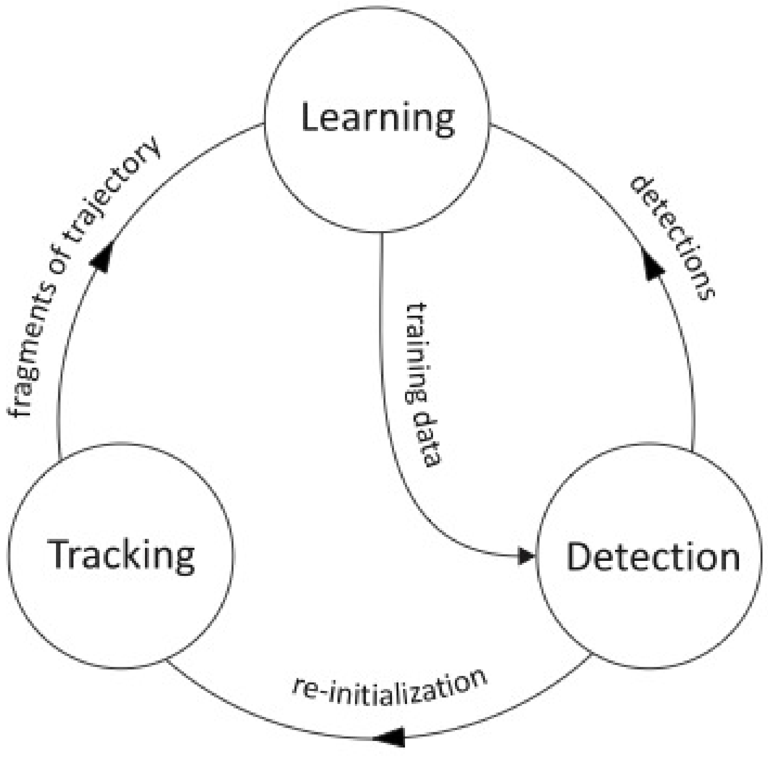


Figure 2.6: High-level view of TLD

Image from [20].

In a recent work, we see the powerful result of combining these two concepts along with *Machine Learning* to produce a brand new approach. Known as **Tracking–Learning–Detection (TLD)** [20], this method uses an independent Detector and Tracker working

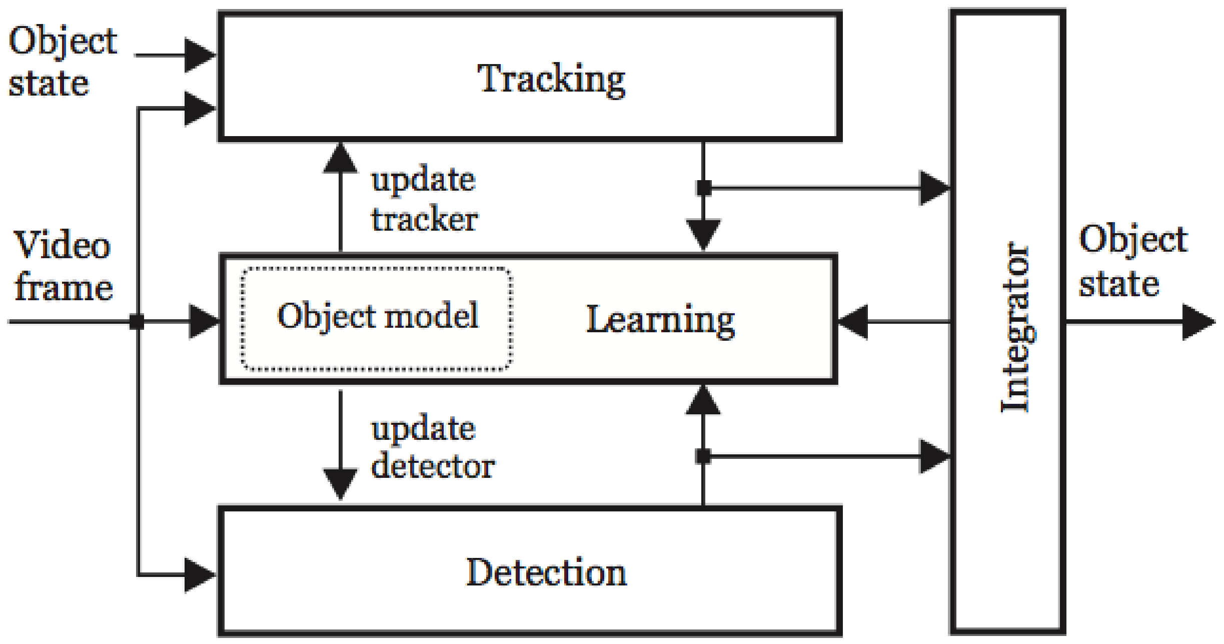


Figure 2.7: Detail view of TLD framework

Image from [20].

simultaneously, but separately Fig. 2.6. The strengths of the Detector (e.g. error recovery) make up for weaknesses in the Tracker. Likewise, strengths of the Tracker (e.g. speed) make up for weaknesses in the Detector. The Learning module, independent of and simultaneous with the previous two, accounts for problems in long-term traking, i.e. model drift and transformations.

### 2.3.1 Object Model

Selection of an appropriate object model is essential. TLD is an adaptive discriminative tracker, which means that an object model can change over time and that it compares tracked objects with their background. Therefore, in TLD, an Object is a data structure containing a collection of normalized, labeled patches.

Positive patches, those depicting the object, are sampled from within the object bounding box, and ordered by time of first detection. The ordering of positive patches is used in certain similarity measures within TLD, effectively giving priority to earlier views of the object. This causes the object detector to behave conservatively when confidence is low. Negative patches

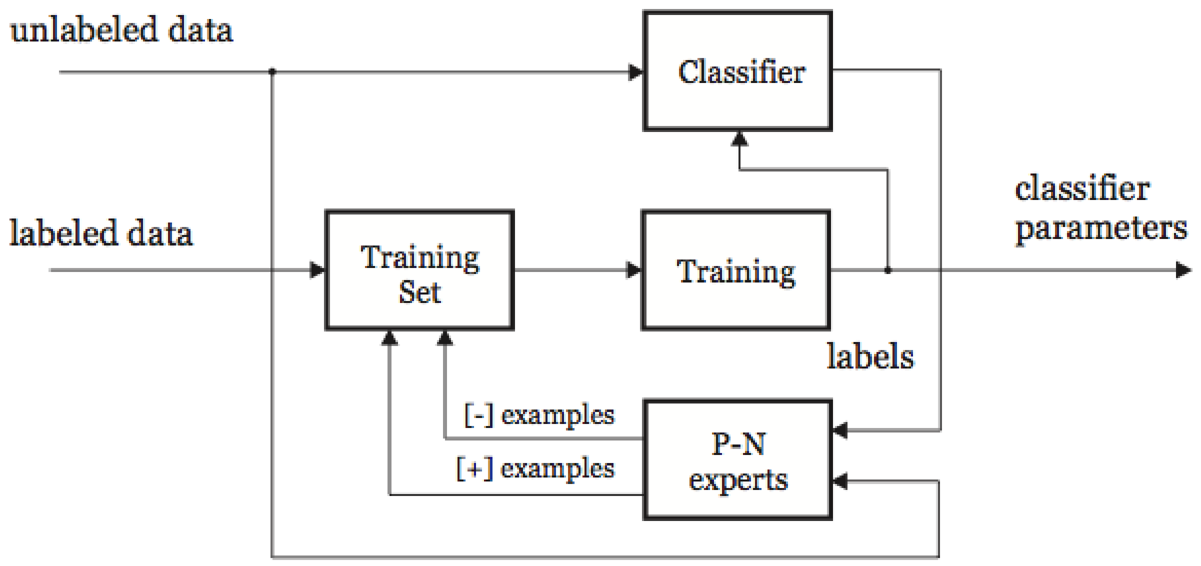


Figure 2.8: Learning process in TLD

Image from [20].

are those containing background, like sky or trees, and/or non-tracked objects. For instance, if the task was Facial Tracking, a negative patch for *Face A* could contain the corresponding person's shoulder or hand, parts of another person, or sections of the image background. Negative patches are sampled from outside the target bounding box and are not ordered.

In order to be adaptive during long-term tracking tasks, the TLD Object Model must be periodically updated. This is handled by the *Learner*, which is covered later.

### 2.3.2 *Tracker*

Object Tracking in TLD is performed by the *Tracker*. Starting with the initial view of the target, it estimates motion using **Median Flow** [21] tracking, extended to include failure detection. Median Flow is based on pyramidal L–K Optical Flow with two levels and 10 x 10 pixel patches for image features. Median Flow tracking contributes the idea of **Forward–Backward Error**, an error measure for inconsistencies in the trajectory of tracked points. When estimating the tracker error, several video frames are reversed, then the trajectory of each feature point is calculated on that short sequence. Features with trajectories that do not match the majority of others, i.e. outliers, are rejected and not used when estimating the frame-to-frame motion of the tracked object. Outliers occur due to tracking failures, usually from a mismatch of one feature to another, similar feature when the correct feature is occluded.

When the Median Flow tracker has low confidence in its motion estimate, usually due to rapid occlusion or rapid motion beyond the system threshold, *Tracker* signals tracking failure by not returning a bounding box for the tracked object on the frame in question. This guards against erroneous, low-confidence motion estimates. Instead of making a best effort attempt, the system relies on the *Detector* to localize the tracked object and reinitialize *Tracker*.

### 2.3.3 *Detector*

Object Detection in TLD is performed by the *Detector*, which does a complete search for the tracked object at every frame because it assumes that all frames are unrelated. It uses a scanning-window grid approach and a three level cascaded binary classifier that is trained online.

First, the detector takes as input the last known bounding box for the object and then generates a large number new bounding boxes ( 50 k, depending on aspect ratio) at different possible scales and positions within the current frame. Each of these new boxes is then converted to a normalized to a patch and then passed to the classifier, which determines whether each box/patch candidate does or does not contain the object. To limit the number of comparisons made to the object model, *Detector* works in a cascade, reducing the number of candidates by a large factor before moving to the next stage. Higher level stages are simple and fast/cheap to compute. Lower levels are more complex and slower or more expensive to compute. At stage one, candidates pass a *patch variance* test, which typically rejects a large number of patches containing large uniform areas, i.e. background regions like sky or road. At stage two, patches pass through an ensemble classifier, which classifies the patch as containing the object if the average of the posterior probabilities (ranging from 0 to 1) resulting from the base classifiers of the ensemble is greater than 0.5, i.e. confidence exceeds

50%. Patches not accepted at stage two (approximately 50) remain undecided and pass to stage three, a Nearest Neighbor classifier that uses the the *relative similarity* metric defined in [20]. After stage three, all candidate patches are classified and labeled. Zero, one, or many matches can result.

Finally, the responses of *Tracker* and *Detector* are integrated with equal weight Fig. 2.7. If neither returns a bounding box, the object is not visible within the frame and trajectory is not updated. Otherwise, the box with highest confidence is used.

### 2.3.4 ***Learner***

The idea of combining tracking with detection has been explored previously, in various forms, but TLD is set apart by the addition of a Machine Learning method called P–N Learning, first proposed in [22]. Output from the tracker is continuously used to improve the detector through online retraining from the *Learner* (Fig. 2.8).

First, *Learner* initializes *Detector* with examples from the first frame, which are provided by two *experts* called P-expert (positive, Px) and N-expert (negative, Nx). At initialization,*Learner* calls upon P+ to generate a set of synthetic positive examples of the target object by sampling patches from the target's bounding box and altering them with affine transforms, warps, and the addition of Gaussian noise. Synthetic examples combined with the actual examples form the starting patches used by *Detector's* cascaded classifier. At the same time, Nx produces negative examples, sampled from outside the target bounding box.

Afterwards, during run-time, Px and Nx strengthen the detector by providing additional examples. At any given iteration, when the target has been located with high confidence, Px generates a new set of synthetic positive examples ( 100). These are sampled from bounding boxes (generated by the scanning grid window method) closest to the target bounding box. At the same time, since the target can only be in one location in a given frame, Nx samples patches from outside the known location of the target, particularly in places that the detector had suggested as a location for the target, but that were not confirmed by the tracker. These patches are given to the classifier as negative examples.

## 2.4 Autonomous Vehicles

During the past decade, there has been a tremendous amount of research devoted to building autonomous vehicles of every type, for every environment. As with most technologies, the primary drivers are military and industrial concerns, followed by commercial and research interests. Autonomous vehicles are grouped by the area they are designed to operate in: sea, air, land, etc. Sometimes, they are grouped together under abbreviation UxV, that stands for Unmanned x Vehicle, with another term substituted for x.

### 2.4.1 Terrestrial Vehicles

Terrestrial Vehicles are the most accessible form of autonomous vehicle, especially due to the prevalence of automobiles in industrially developed regions. Many modern automobile makers (Audi, BMW, Volkswagen) offer vehicles with semi-autonomous systems, like forward-collision warning, lane-departure warning, parking-assist, blind-spot monitoring, adaptive cruise control, and pedestrian detection. Parking assist systems combine a variety of sensors, along with knowledge of a vehicle's physical properties, to maneuver an automobile into an empty parking space. This is done without the intervention of a human driver, all while avoiding static and dynamic obstacles. Systems are available to assist with parallel and perpendicular parking.

The Defense Advanced Research Projects Agency (DARPA) regularly holds a Grand Challenge, a research and development contest in robotics, originally focusing on Unmanned Ground Vehicles (UGVs). Thrun [23] describes the hardware-software framework underlying Stanley, an autonomous driving vehicle from Stanford. Stanley placed first in the 2005 DARPA Grand Challenge, a driving competition held in the Mojave Desert. In 2007, the DARPA Urban Challenge was won by "Boss" (Fig. 2.9), fielded by Tartan Racing from Carnegie Mellon University.

Google is famous for its development and use of self-driving cars for capturing data for the popular Google Maps Web-based application. While employing a human driver for safety

Figure 2.9: 2007 DARPA Urban Challenge winner, “Boss”

Image from: `http://www.tartanracing.org/`

and legal concerns, the cars navigate streets with ease while capturing images for Street View feature of Google Maps.

In [24], the authors present a Computer Vision-based autonomous docking system for the Mars Rover. They use pose estimation algorithms for identifying targets in an unknown environment. In [25] the authors describe an autonomous driving framework that integrates data from multiple sensors such as GPS, odometers, etc. To sense the environment, they employ a combination of radar, LIDAR, and cameras.

### 2.4.2 Aerial Vehicles

Unmanned Aerial Vehicles (UAVs), sometimes simply called *drones* have garnered the greatest attention in recent years, primarily due to their increased deployment for military purposes like surveillance and missile strikes However, myriad other uses exist for UAVs, deployed on a wide variety of platforms. Helicopter-like vehicles (Fig. 2.10) or quad-rotors are useful for remote inspection of infrastructure, like checking bolts and welds on bridges or inspecting the exterior of tall buildings. Fixed-wing, propeller driven vehicles are useful for inspecting pipelines, power lines, etc. Any sort of UAV could be used to check forests or plains for signs of drought and early detection of wild fires. Likewise, any sort of UAV could be useful for monitoring migration patterns of far-ranging land animals.

Figure 2.10: Northrup Grumman MQ-8B Fire Scout

Image from: `http://www.northropgrumman.com`

In recent work, PIXHAWK [26] provides a hardware-software framework for micro-air vehicles that uses data obtained from an inertial measurement unit (IMU) and cameras that are hardware synchronized to provide close coupling.

Several fast food companies, Internet giants like Google and Amazon, and delivery companies, have recently been working with contractors to develop *home-delivery drones* [27, 28, 29]. The technologies involved are mostly ready, with the main hurdles being cost-effectiveness and regulatory concerns from the Federal Aviation Administration regarding safety, privacy, and responsible use.

### 2.4.3 Marine Vehicles

Surface marine vessels, though one of mankind's oldest technologies, remains a continuous locus for development. Take the Saildrone for example [30], (Fig. 2.11). It combines advanced composite materials with a streamlined design to make the most efficient use of thrust generated by the sail. The key to the design is the "sail", which is really more of a wing, positioned vertically, with a tail protruding from the trailing edge. As wind pushes the tail, it automatically turns the wing into the wind, so the wing is *always* positioned to generate thrust. Navigation and communication equipment was custom made with low power consumption and sealed housing to keep out water and resist corrosion. All electronics are powered by solar panels on deck. The triple-hulled design (main hull with two outriggers) is self-righting and fully submersible [31].

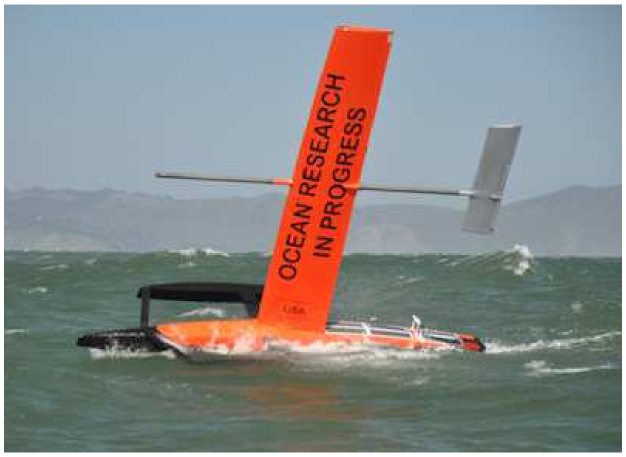


Figure 2.11: Saildrone on the water

Image from: `http://saildrone.com`

Many design components for Saildrone were inspired by similar developments in Autonomous Underwater Vehicles (AUVs), also known as UUVs, or Unmanned Undersea Vehicles. Aside from on-board control systems, UUVs have much in common with underwater ROVs (Remotely Operated Vehicles), both of which are commonly used in research, exploration, and industry. Collectively, UUVs and USVs (Unmanned Surface Vehicles, e.g. saildrone) are known as UMSs (Unmanned Maritime Systems).

Dynamically Positioned Vessels (as discussed in Ch. 2.1) fall under the category of Autonomous Marine Vehicles, to an extent. Though for the majority of the time it would be *nonautonomous* or *semiautonomous*, a DPV's maneuvering capability is intended to be fully automated with little or no human supervision, at least during DP operation.

# Chapter 3
# Problem Description

After analyzing and comparing different types of existing Position Reference Systems, we have observed a niche left unfilled by any current system. The oldest, simplest systems are error prone, not precise enough for modern DP operations. Many are constrained by water depth and/or unsuitable for mobile operations, e.g. pipe laying, moving in formation, collision avoidance. Most of the recently developed systems require lengthy, expensive installations and maintenance, particularly systems that use specialized targets to track reference objects.

In contrast, *SeaVipers* will be an inexpensive alternative. The system will be small, lightweight, and easy to install. It will not require special, prepared targets (e.g. transponders, reflectors) to be installed and maintained on reference objects. *SeaVipers* will be suitable for both mobile and stationary applications in any water depth or lighting condition. It will not be subject to interference from noise, reflective surfaces, direct sunlight, or the radio shadowing effect of large metal structures. Neither will it be susceptible to interference from other *SeaVipers* units operating nearby, either those installed on other vessels, or possible extensions like cooperative, multi-unit installations on one vessel.

*SeaVipers* will operate in high seas and inclement weather, at least as well as existing laser PRS like Fanbeam® or CyScan. Due to it's LWIR camera, it will penetrate rain, fog, and other atmospheric obstructions better than existing optical systems, which only use NIR. It will feature rapid system start-up, requiring only a few minutes at most to learn new reference objects. After that, it will only take a few seconds to acquire the target, or reacquire after target loss.

For this goal, *SeaVipers* must meet the following requirements. It should operate with minimal start-up time, continuously for periods exceeding twenty-four hours, with minimal supervision from the DPO, in moderate wind and wave conditions, in lighting conditions

Table 3.1: Comparison of Optical PRS

| | *SeaVipers* | Fanbeam | CyScan |
|---|---|---|---|
| max range | unknown | 2000 m | 800 m |
| reflectors | no | yes | yes |
| optics | NIR, LWIR | NIR | NIR |
| pan | 360 | 360 | 360 |
| tilt | +/- 90 | +/- 15 | +/- 20 |

from full dark to direct sunlight, and in moderate to heavy precipitation (rain, fog, etc.). To compete with Fanbeam® and CyScan, *SeaVipers* should operate effectively in common DP conditions from a range of at least 500 m to target. Under ideal conditions (calm seas, calm winds, no clouds or precipitation, noon-time sunlight), we expect the max range to be much greater, though this not critical.

# Chapter 4
# System Overview

This section discusses the various components and functions of *SeaVipers* and how each contributes to the operational goal of the system.

## 4.1 Hardware

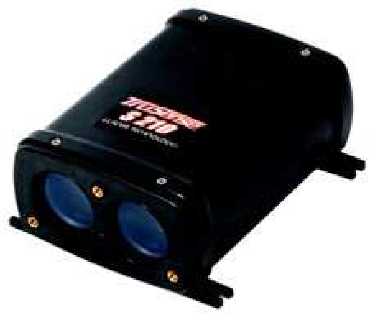
### 4.1.1 Pan/Tilt Unit

For computer vision applications on a mobile, unstable platform, a suitable Pan–Tilt Unit (PTU) is essential. In our application, the purpose of the PTU is to keep the camera and rangefinder aimed at the target. Heading information is obtained from the PTU's internal orientation sensors. This version of *SeaVipers* uses a FLIR PTU-D47, shown in Fig. 4.1a. The D47 was chosen for its high pan/tilt speeds and fine pan/tilt resolution, which are both important features for real-time operations which must respond to the constant motion caused by wind and waves.

### 4.1.2 Rangefinder

To measure range to target, *SeaVipers* uses a LTI TruSense S210 Laser Sensor (Fig. 4.1b), which measures the time-of-flight of Near-Infrared (NIR) light pulses. Operating in the NIR range allows the S210 to operate in all lighting conditions and allows it to penetrate atmospheric precipitation (rain, fog, snow, etc.), which is critical for continuous, long-running

(a) Pan/Tilt Unit


(b) Laser Rangefinder

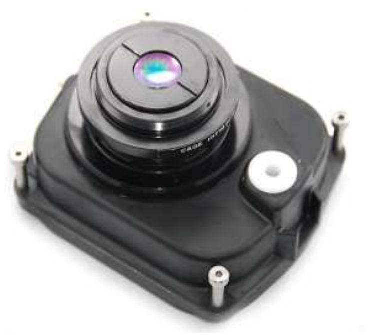

(c) Infrared Camera

Figure 4.1: Hardware Components

(a) FLIR PTU-D47, (b) LTI TruSense S210 Laser Sensor, (c) Raytheon Thermal-Eye 300D operations (in excess of 24 hours) in marine environments. Also, the S210 *does not require reflectors* to obtain measurements, which allows rapid start-up and operation of the system.

#### 4.1.3 Camera

The optical sensor for *SeaVipers* is a 9 Hz Ratheon Thermal-Eye 300D infrared camera. The 300D (Fig. 4.1c) uses an uncooled Barium Strontium Titanate detector, which is unaffected by longterm exposure to direct solar radiation, a valuable feature for continuous, long-running operation at sea. The detector has spectral response to Long-Wavelength Infrared (LWIR) which allows operation in any lighting conditions and allows it to penetrate atmospheric precipitation better than visible-light cameras [32].

#### 4.1.4 Inertial Measurement Unit

To assist the PTU in video stabilization for the camera and aiming the rangefinder, *SeaVipers* uses an Inertial Measurement Unit (IMU) to detect changes in roll, pitch, and yaw (x-, y-, and z-axis movement). We use a 9DOF Razor IMU from SparkFun Electronics, which has nine Degrees of Freedom (DOF), combining a triple-axis gyroscope, accelerometer, and magnetometer. Inertial stabilization allows the tracking system focus on changes in overall position instead of frequent back-and-forth rocking from waves and wind.

### 4.2 Software

*SeaVipers* is built on top of several existing, proven software resources. First, we use the popular OpenCV (Open Computer Vision) library, version 2.4.6 [33]. This open-source library is well documented and provides powerful tools for rapid development of CV programs. Second, we use G. Nebehay's version of the OpenTLD (Open Tracking–Learning–Detection) library [34], an open-source alternative to Z. Kalal's original TLD program [35], and also an alternative to Z. Kalal's version of OpenTLD [36], which depends upon non-free software tools (G. Nebehay's version does not). Finally, for the Graphical User Interface (GUI), we use Qt 5 [37], an open-source, cross-platform application and user-interface framework for the C++ programming language.

The GUI is designed with a simple, uncluttered main-view for ease-of-use and rapid start-up by the DPO. The main-view presents the user with video from the camera, overlaid with bounding-boxes and trajectory info from the tracker/detector. It also contains a top-down diagram of the vessel and its position relative to the target, buttons to start/stop or pause tracking, a summary graph of changes in range and heading, and a summary of the most recent notifications or warning messages. Additional views are available to access detailed graphs, precise configurations and settings, message logs, and system logs.

## 4.3 Operations

Possible use cases include station keeping near an offshore platform (Fig. 4.2), maintaining a predetermined course during pipe laying or cable laying, or moving in tight formation.

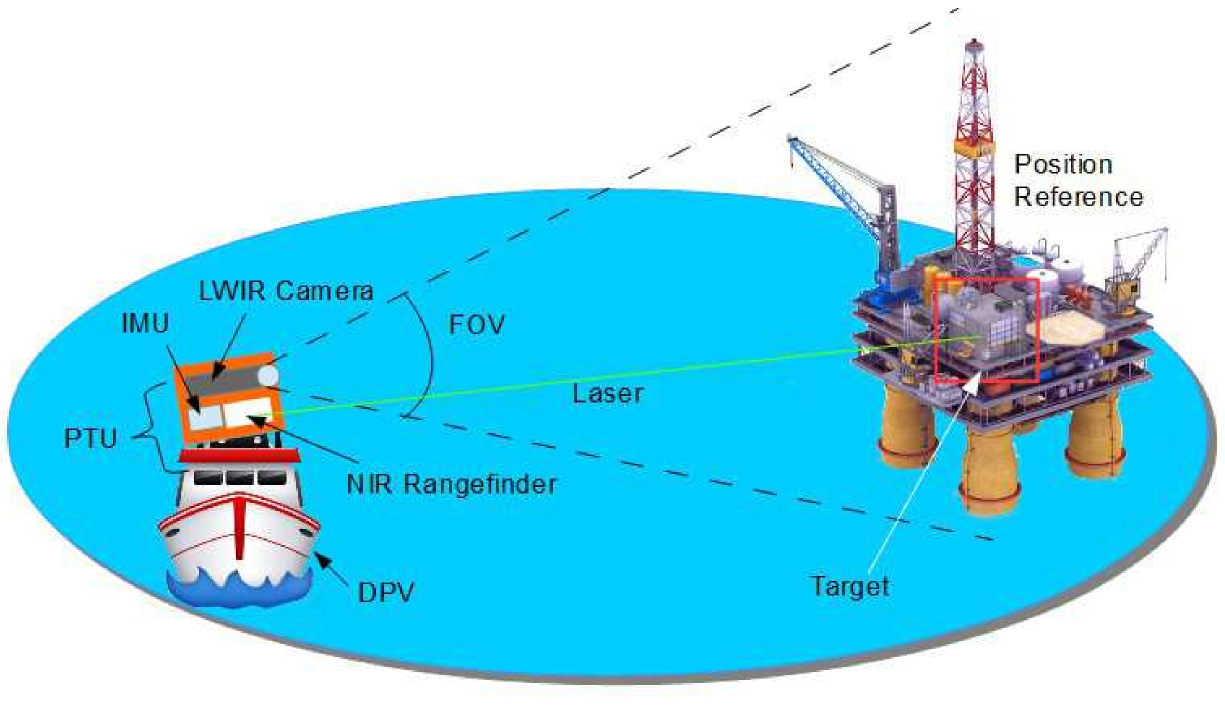


Figure 4.2: Example use case: station keeping near offshore oil drilling platform

*SeaVipers* accesses raw video input from the camera, preprocesses and performs video stabilization, and then feeds video to the target tracker. To engage the tracker, the operator must first manually select a bounding box containing the target. The tracking module begins with a brief "Learning" phase wherein all views of the target are recorded as positive examples. After learning the target model, the tracker begins the detection and tracking phase wherein each incoming video frame is searched for the target.

Whenever the target is found, the tracker reports the target's bounding box to the control module. If the target is found to be in the correct position in frame (detected by detector and near-center), the system is aimed at the target and therefore aligned properly to take measurements. The control module will then request angle-of-heading from the PTU's orientation sensors and range-to-target from the rangefinder. If the the target is in frame, but not aligned (detected, but not near-center), the control module calculates the difference between target position and a good position, then issues an instruction for the PTU to compensate orientation. If the target is not detected, the tracker does not update its bounding box.

On subsequent frames, if the target comes back into view, the tracker can reacquire the target and resume tracking. Finally, if the target cannot be reacquired after a prolonged period, whether due to occlusion, or diminished visibility, or another condition, the system will report target loss (tracking failure) to the DPO. At this point, the DPO can either reinitialize tracking, adjust system settings, or disengage the system.

While tracking operations are going on, the control module regularly updates its own internal model of ship position relative to target, taking measurements from each sensor and modifying its position estimate based on its confidence in each sensor's respective input. At regular intervals, the control software module reports heading and range to the DP Control System.

# Chapter 5
# Evaluation of Prototype

## 5.1 Performance Evaluation

Table 5.1: Test Sequence Summary

| name | frames | light | distance |
|---|---|---|---|
| Lab | 1431 | bright | 0 – 10 m |
| Office | 131 | bright | 0 – 5 m |
| Car 1 | 1832 | norm | 0 – 35 m |
| Car 2 | 1599 | norm | 5 – 20 m |
| Truck | 1773 | norm | 0 – 15 m |
| Crane | 2249 | bright | 150 – 175 m |

*SeaVipers* was evaluated over six sequences (Table 5.1) at different ranges and lighting conditions. The first two, Lab (Fig. 5.1a) and Office (Fig. 5.1b), were conducted indoors, with artificial light, at short range. The next four were conducted outdoors, facing a parking lot from mid-afternoon to early evening (Fig. 5.1 c, d, e, f). Of the frames where the tracker is active, each frame of each sequence was labeled by hand.

The object tracking task can be reduced to a binary classification task, with each individual frame as an input. Every frame receives one of four labels: True Positive, True Negative, False Positive, or False Negative. A **True Positive (TP)** classification is one wherein the tracker correctly identifies the location of the target by providing a bounding box that contains an image of the target. A **True Negative (TN)** classification is one where the tracker correctly determines that the target is not in view, either due to an occlusion or the target having gone out of frame. **False Positive (FP)** denotes instances where the tracker provides a bounding box that *does not* contain the target. Finally, **False Negative (FN)** represents instances where the tracker fails to locate the target even though it is clearly in view within the frame.

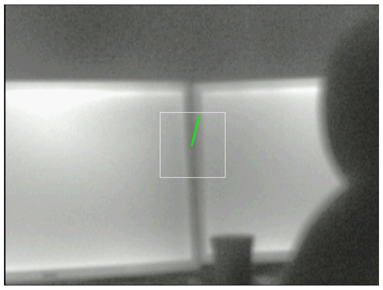
Performance is summarized by two measures: Precision, the probability that the tracking system finds a target object (5.1), and Recall, the probability of the tracking system *correctly* indicates the location of a target object (5.2).

$$Precision = TP/(TP + FP) \tag{5.1}$$

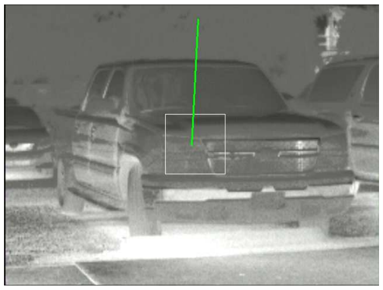
$$Recall = TP/(TP + FN) \tag{5.2}$$

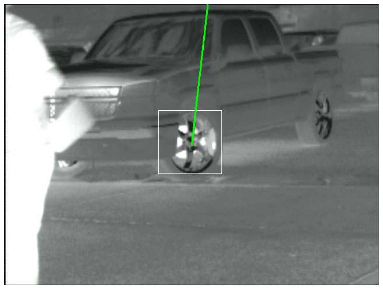
## 5.2 Evaluation Results

(a) Lab


(b) Office

(c) Car 1


(d) Car 2


(e) Truck

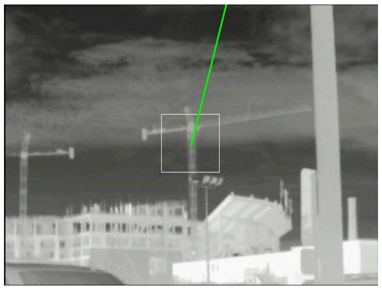

(f) Crane

Figure 5.1: Tracking Sequences

Target bounding box shown in white.

As expected, based on the performance of the original implementation of TLD [20] and the implementation of OpenTLD [38], the tracking implementation for *SeaVipers* shows high recall and precision overall. Table 5.2 shows high precision due to the very low incidence of false positives, with the exception of the Car 1 sequence. System lag during a quick, z-axis

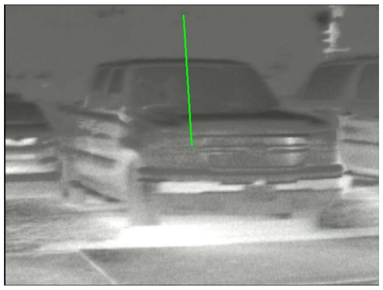
Table 5.2: Tracking Performance by Sequence

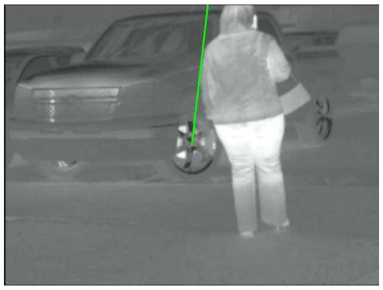
| Sequence | Frames | FP | FN | TP | TN | Recall | Prec. |
|---|---|---|---|---|---|---|---|
| Lab | 742 | 0 | 53 | 689 | 0 | 0.929 | 1.000 |
| Office | 58 | 0 | 0 | 58 | 0 | 1.000 | 1.000 |
| Car 1 | 1463 | 232 | 272 | 561 | 398 | 0.673 | 0.707 |
| Car 2 | 1271 | 3 | 117 | 1142 | 9 | 0.907 | 0.997 |
| Truck | 1288 | 0 | 259 | 940 | 89 | 0.784 | 1.000 |
| Crane | 2216 | 1 | 90 | 1915 | 210 | 0.955 | 0.999 |
| Average | | | | | | 0.875 | 0.951 |

rotation prevented the PTU from reorienting. When operation resumed, the system had dropped several incomming frames, leaving the tracker disoriented, at which point it began falsely identifying a similar object as the target. This sequence also had a high incidence of false negatives, primarily due to motion blur.

The second sequence, Office, has no movement with a target at extreme close range. This was done to demonstrate the stability of the system. All other sequences featured stationary targets tracked by a moving vessel, with translations in the xy-plane and rotations on the z-axis (yaw). Car 2 and Truck also feature roll and pitch (x-axis and y-axis rotations), which the PTU adjusted for with little difficulty.

Observation of the experiments and analysis on the data collected shows that the primary cause of tracking error for *SeaVipers* is rapid motion, beyond what the PTU can account for, especially when the target goes off frame. (See Fig. 5.2 for examples.) The IMU, which would have corrected for this limitation, was only partially implemented. The PRS was also


(a) Car 2 / Motion Blur / FN


(b) Truck / Occlusion / FN

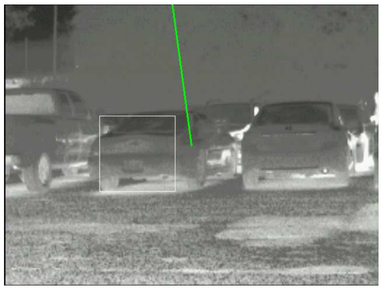

(c) Car 1 / Similar Object / FP

Figure 5.2: Examples of Tracking Error

Sequence / Description / False Negative (FN) or False Positive (FP)

limited by its incomplete implementation of OpenTLD, which did not take full advantage of the *Learning* aspect of TLD, resulting in overly rigid target models.

Occlusions, full and partial, were an unexpected source of error. When the object *Detector* in TLD failed due to an occlusion, the *Tracker* should have predicted the location, at least when the target and PRS were both stationary. This suggests that our implementation underutilizes the predictive capacity of the underlying tracking approach. In contrast, the system did perform as expected during scale changes, showing zero tracking errors due to scaling.

Environmental factors played only a small role in this evaluation. All tests were done in clear weather with no dust or precipitation and bright or moderate, direct sunlight. However, it is worth noting that a small number of errors occured ($< 10$) due to sudden changes in illumination from passing clouds. This may need to be accounted for in future development.

Consistent with nature of TLD, there was low incidence of false positives, excpet for the incident noted during the Car 1 sequence. Likewise, there was a high number of true negatives in Car 1 and Crane, both having long periods where the target is off frame. This is encouraging, as prompt and correct notification of target loss is important in DPS, either to prompt the PRS to attempt reacquisition or to alert the DPO to intervene.

Relating to target reacquisition, we noted that our PRS rapidly picks up lost targets as they become un-occluded or come back in frame, usually within 2 or 3 frames.

# Chapter 6
# Conclusion

This document describes the design and proposes the development of an Optical Position Reference System for use with Dynamic Positioning Systems. The proposed system, called "Computer Vision and Inertial Position Reference Sensor System" (CVIPRSS, *SeaVipers*), uses TLD to perform Object Tracking in a dynamic maritime environment, providing continuous position estimates to the Dynamically Positioned Vessel in relation to the tracked reference object. Position estimates are derived from information gathered from a Far Infrared Camera, a Near Infrared Laser Rangefinder, and 9DOF Inertial Measurement Unit, all mounted atop a compact Pan/Tilt Unit. Unlike some current generation PRSs, *SeaVipers* gathers information without a long setup time or expensive, prepared targets.

Preliminary evaluation of the *SeaVipers* prototype yields promising results while suggesting further improvements on the system's design and implementation.

# References

[1] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *AAAI/IAAI*, 2002, pp. 593–598.

[2] Z. Kalal, J. Matas, and K. Mikolajczyk, "Online learning of robust object detectors during unstable tracking," in *Proc. IEEE International Conference on Computer Vision Workshops (ICCV'09)*, Kyoto, Japan, Sept. 2009, pp. 1417–1424.

[3] B. D. Lucas and T. Kanade, "An iterative image registration technique with and application to stereo vision," in *Proc. International Joint Conference on Artificial Intelligence (IJCAI'81)*, 1981, pp. 674–679.

[4] E. Grabianowski. (2014, Jan.) How self-parking cars work. [Online]. Available: http://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/self-parking-car.htm

[5] *Guide for Dynamic Positioning Systems*, American Bureau of Shipping Std., Nov. 2013.

[6] International Marine Contractors Association. (2004, Nov.) A review of the artemis mark v positioning system. imcam174.pdf. [Online]. Available: http://www.imca-int.com/media/73407/

[7] ——. (2001, Apr.) RadaScan microwave radiation sensor for dynamic positioning operations. imcam209.pdf. [Online]. Available: http://www.imca-int.com/media/73641/

[8] ——. (2003, Nov.) A review of marine laser positioning systems. imcam170.pdf. [Online]. Available: http://www.imca-int.com/media/73380/

[9] R. Szeliski. (2010, Sept.) Computer vision: Algorithms and applications. Szeliski-Book_20100903_draft.pdf. [Online]. Available: http://szeliski.org/Book/

[10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 2564–2571.

[11] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, pp. 91–110, Jan. 2004.

[12] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[13] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 778–792.

[14] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 1, pp. 105–119, 2010.

[15] K. Fukunaga and L. D. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” *IEEE Trans. Inform. Theory*, vol. 21, pp. 32–40, 1975.

[16] S. Mori, H. Nishida, and H. Yamada, *Optical character recognition.* John Wiley & Sons, Inc., 1999.

[17] G. R. Bradski. (1998) Computer vision face tracking for use in a perceptual user interface. camshift.pdf. [Online]. Available: http://www.dis.uniroma1.it/~nardi/Didattica/SAI/matdid/tracking/

[18] B. K. Horn and B. G. Schunck, “Determining optical flow,” in *1981 Technical Symposium East.* International Society for Optics and Photonics, 1981, pp. 319–331.

[19] B. D. Lucas, “Generalized image matching by the method of differences,” Ph.D. dissertation, Carnegie-Mellon University, Pitttsburgh, Pennsylvania, 1984. [Online]. Available: http://www.ri.cmu.edu/pub_files/pub4/lucas_bruce_d_1984_1/

[20] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-Learning-Detection,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 34, pp. 1409–1422, 2012.

[21] ——, “Forward-Backward error: Automatic detection of tracking failures,” in *Proc. IEEE International Conference on Pattern Recognition (ICPR’10)*, Istanbul, Turkey, Aug. 2010, pp. 2756–2759.

[22] Z. Kalal, J. Matas, and K. Mikolajczyk, “P-N learning: Bootstraping binary classifiers by structural constraints,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR’10)*, San Francisco, USA, June 2010, pp. 49–56.

[23] S. Thrun, “Toward robotic cars,” *Commun. ACM*, vol. 53, no. 4, 2010.

[24] D. Tsai, I. A. D. Nesnas, and D. Zarzhitsky, “Autonomous vision-based tethered-assisted rover docking,” in *IROS*, 2013, pp. 2834–2841.

[25] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, “Towards a viable autonomous driving research platform,” in *Intelligent Vehicles Symposium*, 2013, pp. 763–770.

[26] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, “Pixhawk: A system for autonomous flight using onboard computer vision,” in *ICRA*, 2011, pp. 2992–2997.

[27] Amazon PrimeAir. [Online]. Available: http://www.amazon.com/b?node=8037720011

[28] (2012) Burrito bomber: The worlds first airborne mexican food delivery system. [Online]. Available: http://www.darwinaerospace.com/burritobomber

[29] L. O’Connor. (2014, Mar.) Drugstore delivery drones are coming to san francisco. [Online]. Available: http://www.huffingtonpost.com/2014/03/18/drugstore-drone-san-francisco_n_4989375.html

[30] A. Fisher. (2014, Feb.) The drone that will sail itself around the world. [Online]. Available: http://www.wired.com/2014/02/saildrone/

[31] (2013) Saildrone. [Online]. Available: http://saildrone.com/

[32] FLIR. (2014) Seeing through fog and rain with a thermal imaging camera. metrological effects of fog & rain upon IR camera performance. FOG APP_STORY_ENG-Asia LR.pdf. [Online]. Available: http://www.flir.com/cvs/apac/en/technotes/

[33] OpenCV Development Team. (2013, July) OpenCV API reference— OpenCV 2.4.6.0 documentation. [Online]. Available: http://docs.opencv.org/2.4.6/modules/refman.html

[34] G. Nebehay. (2011, Nov.) OpenTLD. [Online]. Available: http://www.gnebehay.com/tld/

[35] Z. Kalal. (2013) TLD — Zdenek Kalal. [Online]. Available: http://info.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html

[36] Z. Kalal *et al.* (2011) Home * zk00006/OpenTLD wiki * GitHub. [Online]. Available: https://github.com/zk00006/OpenTLD/wiki

[37] Digia PLC. (2013, July) Qt 5 — Qt Wiki — Qt Project. [Online]. Available: http://qt-project.org/wiki/Qt_5.0

[38] G. Nebehay, "Robust object tracking based on Tracking-Learning-Detection," Master's thesis, Austrian Institute of Technology, Vienna, Austria, 2012. [Online]. Available: http://gnebehay.github.io/OpenTLD/gnebehay_thesis_msc.pdf

# Vita

J. Erdman began college through the Advanced Academy of Georgia, a residential, early entrance program at the University of West Georgia. There, they earned a BS in Computer Science in 2010. Afterward, they were admitted to the PhD program in Computer Science at the University of Texas at San Antonio, where they studied for two years. Since 2012, they have been pursuing a PhD in Computer Science at Louisiana State University.