8-19-2024

UAV-Based Tracking and Following of Railroad Lines

Keith Michael Lewandowski
University of South Carolina

UAV-Based Tracking and Following of Railroad Lines

by

Keith Lewandowski

Bachelor of Science
University of South Carolina 2023

Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in

Mechanical Engineering

College of Engineering and Computing

University of South Carolina

2024

Accepted by:

Nikolaos Vitzilaios, Director of Thesis

Dimitris Rizos, Reader

Ann Vail, Dean of the Graduate School

Abstract

Given the pivotal role of the railroad industry in modern transportation and the potential risks associated with track malfunctions, the inspection and maintenance of railroad tracks emerges as a critical concern. While existing solutions excel in performing accurate measurements and detection, they often rely on large, expensive, and time-consuming platforms for inspections. This project, however, seeks to solve the same problem with the use of an unmanned aerial vehicle (UAV), significantly reducing time and cost while maintaining detection capabilities. In particular, this solution is ideal for large-scale, high-level inspections following major events such as floods [6], hurricanes [7] or earthquakes [29]. In such cases, UAVs offer a more efficient solution. Moreover, UAVs can still fulfill many additional inspection needs achieved by current platforms. Hence, this project focuses on developing, implementing, and testing a fully functional, vision-based, autonomous track-following system for UAVs. The creation of a cutting-edge track detection algorithm, TrackNet, is used to identify and interpret railroad tracks from the video stream of an onboard camera. This system is then seamlessly integrated with a customized DJI Matrice 100 UAV to detect and follow railroads in real-time. Notably, this system operates independently of external sensors such as GPS, thanks to its utilization of advanced computer vision techniques. Two distinct approaches utilizing differing camera configurations were developed, tested, and compared. Both systems were found to successfully detect and follow railroad tracks 300 meters in length containing curved and straight sections. The first approach required a forward-facing camera and detected the vanishing point of the track as a control reference. The second approach required a

downward-facing camera and detected the center line of the track to be used as a control reference. These two systems were developed and improved to achieve a average track position errors of 1.9766 meters and 2.0342 meters for the forward-facing approach and the downward-facing approach respectively. Utilizing this system, a UAV can autonomously detect and follow railroad tracks, establishing the fundamental framework upon which various inspection algorithms can be developed to suit their specific applications.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The railroad industry plays a pivotal role in the global transportation network, facilitating the movement of cargo, passengers, and supporting local economies [5] [1]. Despite their significance, railroads can pose substantial risks if not adequately maintained [14]. This maintenance must address two types of track deterioration: the gradual wear from continuous usage and major obstructions resulting from specific incidents [6] [7] [29]. Current methods of track maintenance primarily rely on manual methods or automated track geometry vehicles. Both of these methods have advantages, but are also lacking in several ways. Traditionally, railroad tracks have been inspected manually, by inspectors walking along the tracks or riding some type of high-rail vehicle ([19]). Although these methods are very common, they are not completely reliable, are labor-intensive, are time-consuming, and subject inspectors to hazardous environments. Additionally, even when utilizing high-rail vehicles, the maximum inspection speed is around 5 km/h ([19]). For this reason, it is widely accepted that a superior method of track inspection is the utilization of automated track inspection vehicles to measure track and rail geometry. These platforms utilize a host of non-destructive evaluation (NDE) technologies to identify rail surface and track geometry defects ([19]). The primary limitations of such techniques, however, are their speed and their cost. The current systems are capable of performing inspection at around 15 km/h, but also require significant time for deployment and cause track shutdowns for inspection ([19]). Additionally, the average cost of a single track inspection vehicle is around $8.1 million to purchase or $2.2 million annually for a

service contract ([19]). Additionally, these platforms are effective in detecting small defects caused by long-term wear, but they are less efficient at addressing the second type of deterioration induced by major destructive events. The existing technology, due to its time requirements, unnecessary precision, and reliance on the track's viability, is ill-suited to meet the demands of such scenarios. An innovative solution proposed to address this challenge involves the utilization of Unmanned Aerial Vehicles (UAVs) for track inspection [15]. Although current systems are capable of much higher detail of inspection when compared to UAVs, any reduction in their need would allow for significant savings. UAVs are capable of performing many of the same types of inspection as these inspection vehicles at a fraction of the cost (around $5,000) and at least the same speed (at least 14.4 km/h) without the need for track shutdown or lengthy deployment time. UAVs offer the capability to traverse large sections of track, identifying major obstructions at a reduced cost. Moreover, their airborne nature allows them to conduct inspections without being hindered by the obstacles themselves.

The approach to UAV-based track inspection outlined in this paper consists of track detection and track following. In Chapter 2 each of these processes are described and the current literature is evaluated. The system developed, utilizes only an onboard camera as the primary sensing apparatus for high-level flight controls enabling its operation in GPS denied environments such as tunnels or remote tracks. Because of this, the track detection system needed to rely solely on computer vision and associated techniques. Such approaches found in the literature were split into track region detection, track rail detection, and compound region-rail detection. Depending on the type of track detection used determines the type of track following approaches required. These methods are explored and evaluated, informing the proceeding development. Chapter 3 consists of the manuscript of a publication produced outlining the track detection system, TrackNet, utilized onboard the final UAV [13]. TrackNet

was informed by the literature and developed a compound region-rail approach to track detection where first the track region is located, then the rails are identified within this region. The last stage of TrackNet then is to interpret the detected rails. The form of such interpretation is dependant upon the physical position of the camera onboard the UAV and therefore a division in approach is taken. This division is further explored in the first sections of Chapter 4, namely outlining an approach based on a forward-facing camera and one based on a downward-facing camera. Additionally, this chapter continues by explaining the usage of this interpreted data in high-level flight controls and how these interact with the DJI Matrice 100's built in controls. Finally, these two approaches underwent a series of tests and revisions that are explored in Chapter 5. The results of these tests revealed the differences in each of the two approaches and these are discussed in Chapter 6 along with the general findings of these experiments.

1.1 Contributions

This thesis proposes and demonstrates a novel method for railroad track inspection utilizing a UAV system. The goal of this system is to provide the foundational track detection and following techniques that can be utilized for any number of specific inspection applications. Specifically, the contributions of this work are:

- The development of a track detection system (TrackNet) that is a compound region-rail approach utilizing state-of-the-art techniques for both region detection and rail detection in a way not yet seen in the literature.
- The invention of a novel line chaining method for rail identification as the final step in the TrackNet system.
- The implementation of a trajectory approach to track interpretation for a downward-facing camera angle, a method sparsely present in the literature.

- The first implementation of a track following system based on the downward-facing camera angle.
- The first series of thorough experiments validating the implementations of two track following systems, one based on a forward-facing camera orientation and the other a downward-facing orientation.
- A comparison of the two track following systems (forward-facing and downward-facing) and an outline of the strengths and weaknesses of each.

The track detection portion of this work has been published and presented at the 2024 Joint Rail Conference and the remainder is currently being submitted to be published in a journal.

Chapter 2

Review of Literature

This section will outline the techniques and approaches explored previously in literature related to autonomous track following. The review will follow a similar structure to that of the developed system, namely track detection then track following. Although this general approach is somewhat present in the literature, a fully autonomous track following UAV system has yet to be realized. The current state of the literature mostly falls into one of two cases. On the one hand, many researchers have begun development on effective and sophisticated track detection techniques utilizing the state-of-the-art computer vision algorithms, but these techniques are only tested in theory or on image datasets, not implemented into any real UAV systems [18] [24] [3] [23] [16] [22] [4] [12] [2] [26] [8] [27]. This is the state of the majority of the literature, however some attempts have been made at UAV implementations of such techniques. Here though there is a clear reduction in sophistication of the computer vision techniques used and often also a simplified testing situation such as simulation or mock-up sections of track [17] [11]. The state of this literature shows a clear gap between the superior techniques developed for track following and an actual implementation of these techniques onboard a UAV. The goal of the final system is to fill this gap by identifying robust track detection algorithms, implementing them within a UAV control system, and testing their efficacy live in the field.

2.1 Track Detection

The system presented in this paper faces a significant constraint: it relies solely on an onboard camera system as the sensor for calculating high-level controls. Consequently, the system must process video input and generate commands to guide the UAV along the track. To achieve this, determining the UAV's position relative to the track is imperative, necessitating the identification of the track within each frame of the video, this process is called track detection. In the literature, this task is tackled through various approaches, broadly categorized into three groups: region detection, rail detection, and compound region-rail detection. Each approach offers distinct advantages and drawbacks.

2.1.1 Region Detection

The first approach, region detection, operates on the assumption that the area surrounding the tracks exhibits consistent visual features, including common elements such as rails, rail ties, gravel, concrete, etc. Although the methods for region detection vary significantly in complexity, there tends to be some level of processing before and after the primary method of detection.

The processing performed before detection, preprocessings, typically consists of simple alteration of the image properties such as a conversion to HSV (hue, saturation, and value) space ([22]), altering the brightness, thresholding, or blurring. The actual techniques performed in this stage vary from case to case as they are highly dependent on the processes that follow. For instance, [24] utilizes a simple linear iterative clustering algorithm to segment the image into "superpixels". This is beneficial in increasing the speed and accuracy of the proposed classification algorithm that follows and an example of this can be seen in Fig. 2.1.

After this initial preparation has been done the main detection algorithm begins. A simple and explanatory example of a region detection process is in [22]. Here, the

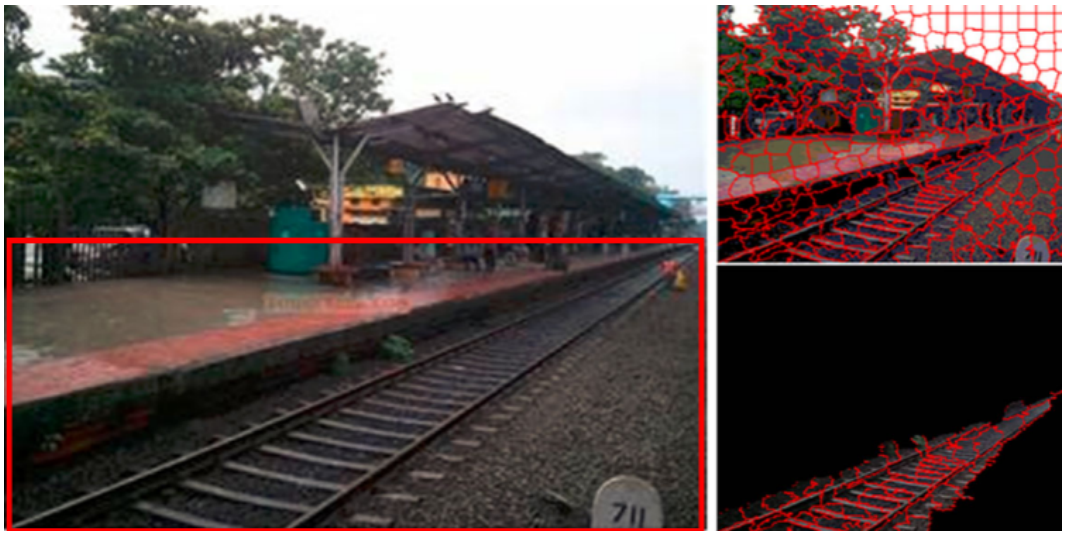


Figure 2.1 Example of preprocessing method for region detection. The researchers in [24] divide the image into 'superpixels'.

researchers identify a particular HSV value as being common to railroad regions in the dataset they were using. This can be seen in Fig. 2.2 as the region of consistent color around the tracks. The researchers then simply isolate the region of the image that contains values within this range and assume that is the track region. Simple methods such as this are highly efficient, but sacrifice robustness as they often rely on features that are not guaranteed to be consistent for all tracks. In the example of [22], it is possible that the specific HSV value identified is particular to the environment and type of track in their dataset. If applied to a broader case, accuracy in detection would be likely to decrease. Other approaches attempt to increase robustness utilizing machine learning algorithms to find and extract deep features in images of railroad tracks. In [24] a support vector machine (SVM) is developed to classify each superpixel in the image as either being part of the railroad or not. This method utilized a TF-IDF like transform to improve the discriminative power of feature recognition. The final classified superpixels are highlighted and seen in Fig. 2.1.

Another approach to identifying the railroad region is through segmentation algorithms. Similar methods are often used in the field of object identification ([2], [27]),

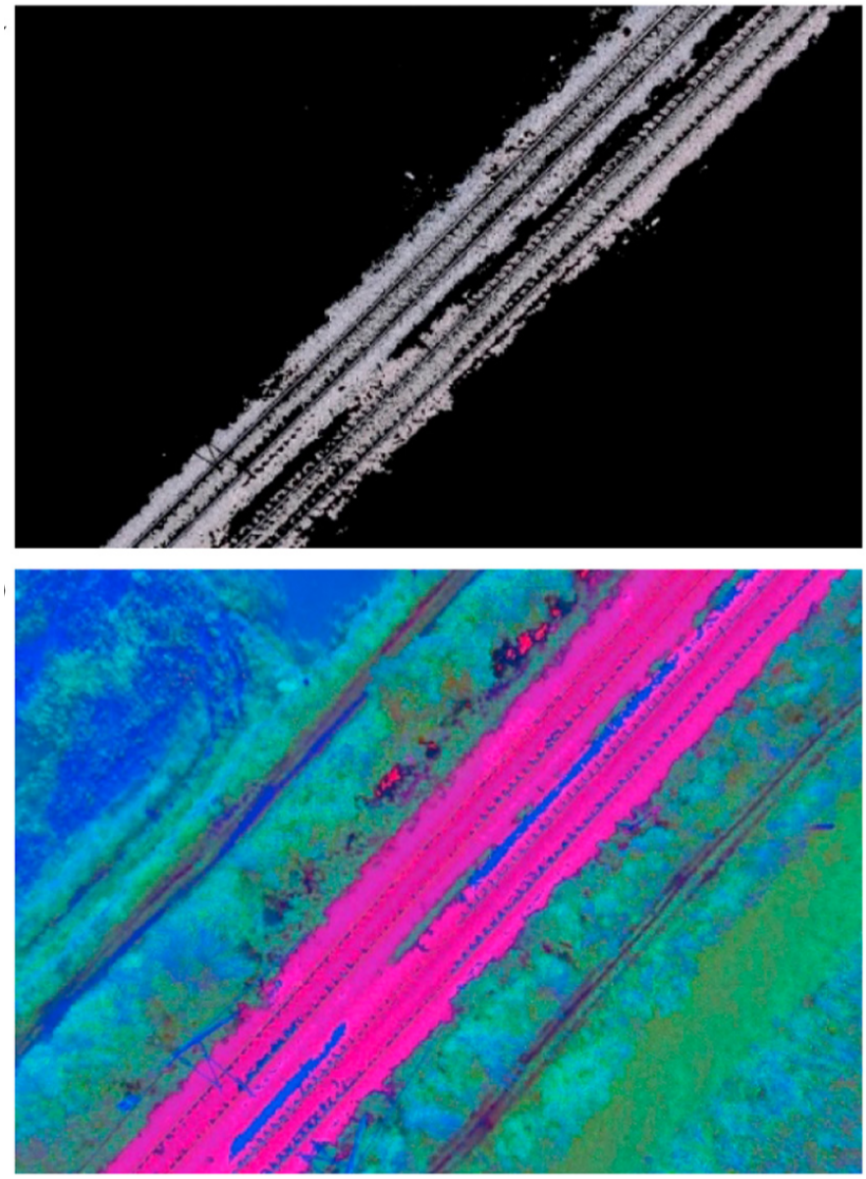


Figure 2.2 Region detection through HSV value isolation [22]

but here they typically utilize some type of machine learning to produce a silhouette-like outline (mask) of the track region. In [26] a semantic segmentation network is developed and called RailNet. An example of the masks produced using this method can be seen in Fig. 2.3. This algorithm uses ResNet50 as the backbone for feature extraction and a fully convolutional network for segmentation. The network is trained on 2500 images and shown to demonstrate superior accuracy for this application than

other, common segmentation networks such as Mask-RCNN, SegNet, DeconvNet, and FCN. These types of systems have a significantly higher accuracy and robustness due to the utilization of machine learning, but also require more computational power that may be difficult to implement onboard a UAV.

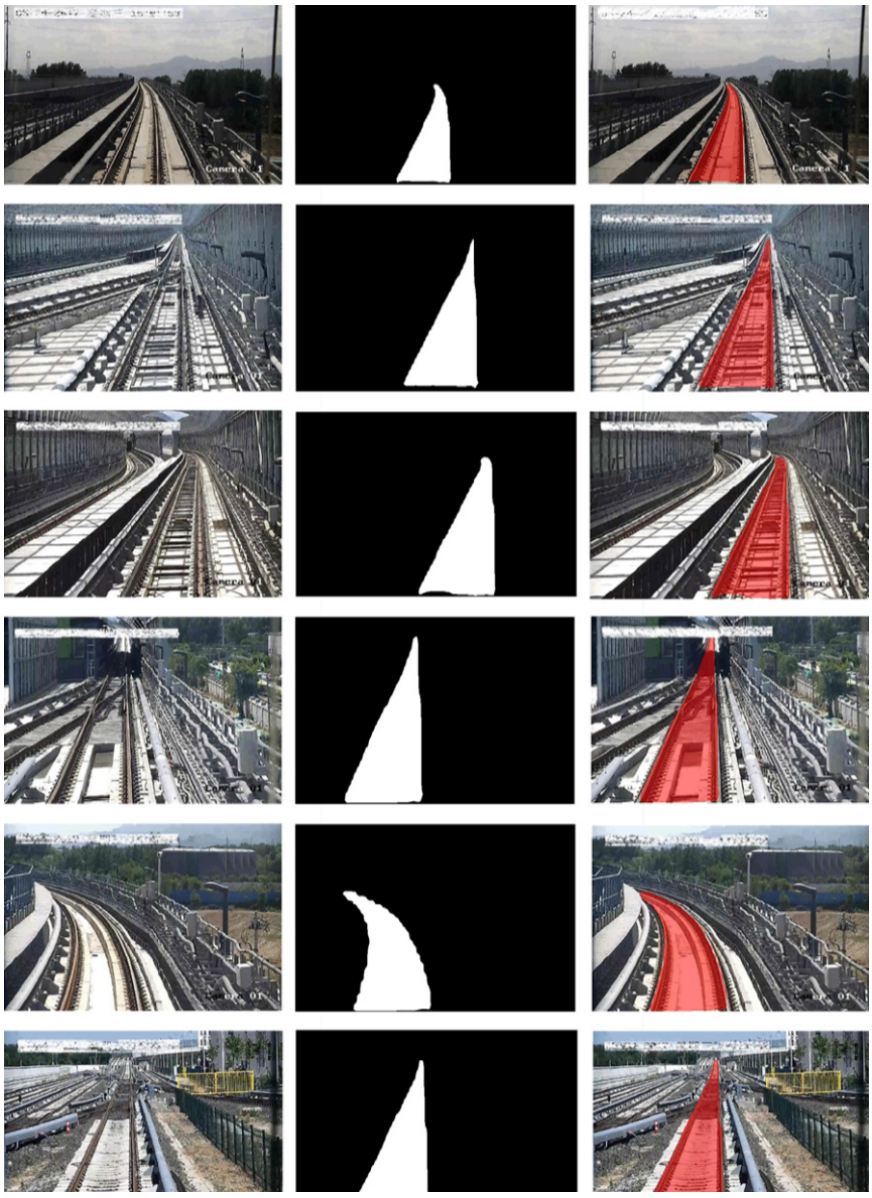


Figure 2.3 Sever examples of RailNet system from [26] conducting region detection through track segmentation.

The framework of region detection has advantages because it isolates the entire

region of the track as opposed to simply a feature of the track. This inclusion of the full track area preserves extra, useful information that could be utilized in later processes. For instance, it designates a defined portion of the image where the search for obstacles can take place. This method for detecting tracks is therefore more relevant as any number of inspection algorithms can be performed once the full area of track has been identified.

2.1.2 Rail Detection

The second computer vision method of detecting rails is similar to the first in that it searches the image for a specific feature common to railways. Instead of searching for the entire region of the track, however, this method chooses to isolate one feature of the track region, namely the rails. Rails are often chosen as the feature to identify due to the long, harsh lines they create visually. These lines can easily be distinguished from the area surrounding them, allowing the position of the rails to be determined. Also, the location of the rails provides the most useful information of any feature in the track region. They are an obvious identifier of the direction the track is traveling. There are a variety of approaches that have been explored in attempt of detecting these lines, however, one methodology surpasses the other in terms of efficacy and simplicity. This method can be seen in [16], [2], [11], [4] and has two basic steps: edge detection and line detection.

Edge detection is the process by which sharp changes in intensity or color (edges) are isolated within an image. Edge detection can be achieved through a variety of methods, but most include applying a convolutional kernel to the image. This field has been well studied and the most common edge detection algorithms that have been applied are the Laplacian method, the Canny method, and the Sobel method [16], [2], [17][11], [4]. Different researchers have determined that some of these methods are better than others for the railroad application, however, their conclusions tend to

differ. In [4] a comparison of each of these methods was performed and the researchers concluded, "after analysis of the different filters, it can be concluded that the most suitable edge detection algorithm for railway detection is the Sobel filter". Each of the methods in this comparison can be seen in Fig. 2.4 and it seems clear why they may have come to such a conclusion. On the other hand, [16] claims that "Canny method is less susceptible to noise and for this reason it is preferred" when comparing the three methods. Finally, the same comparison determined Laplace to be the better method in [17] stating, "the results show that the Laplacian filter performs the best noise filtering and edge detection compared to the Sobel and the Canny methods". These vastly varying conclusions in the literature seem to suggest that each method is capable of supplying effective edge detection and the nuances of the solution's approach are important to consider when deciding which method to utilize. No matter the method that is used, it is almost never utilized in a vacuum. Image preparation is often conducted to improve the output of the edge detection algorithm enhancing the results of the edges detected. Some such preparation procedures include blurring ([2], [17], [4]), thinning ([17], [4]), closing ([11]), opening, erosion, dilation, and thresholding ([17], [4]). These techniques are adjusted to best suit each specific approach and are performed before or after the edge detection algorithm.

After the edges have been detected, the algorithm must determine which edges are part of the rails and which are not. Because the rails are distinguished by being long, uninterrupted lines, line detection is a natural candidate for this step. There are a variety of methods utilized to achieve line detection, but the algorithm used most in the literature is some variety of the Hough Transform, most typically the Probabilistic Hough Transform.

Similarly to the edge detection algorithms, there are many different means of processing the edge-identified image, prior to the line detection algorithm, improving results. These techniques are often the same as in edge detection section, i.e.,

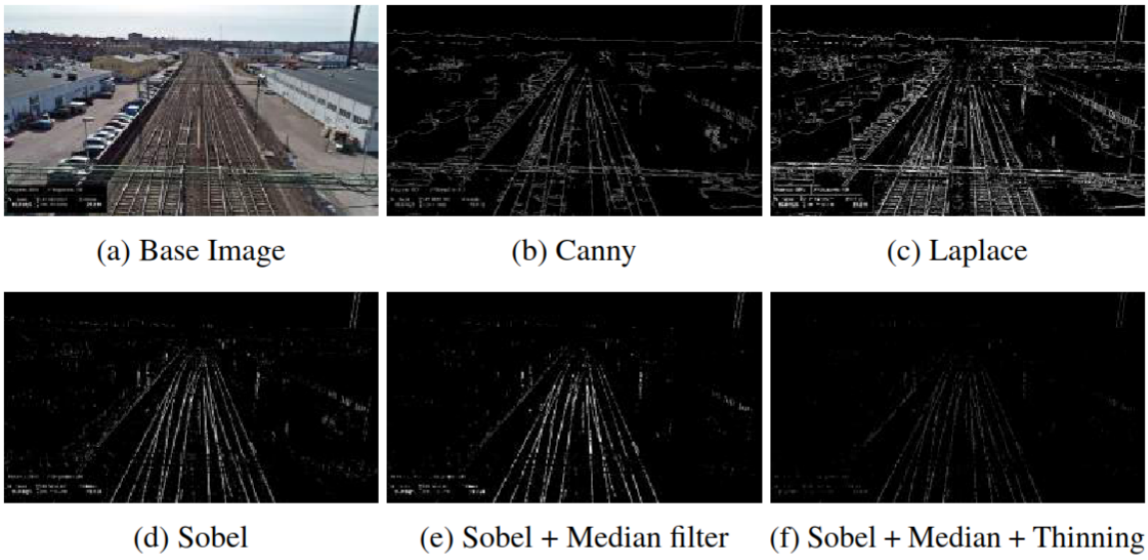


Figure 2.4 Example comparison of differing edge detection techniques from [4].

blurring, thinning, thresholding etc. For instance, in [17] after edge detection is performed, a thresholding process is used to "sharpen" and "enhance the intensity of the strong edges" before implementing a Probabilistic Hough Transform. Another typical processing technique occurs after lines have been identified and seeks to filter the lines to find those which are most likely to be rails. After receiving the lines identified by the transform, the researchers in [17] filter out any lines that are approximately horizontal or vertical. In [16] a line processing system is developed that is comprised of two clustering blocks and one scoring block. This system categorizes the identified lines based on their ρ and θ values into clusters and then calculates a score of how likely that cluster is to be a rail. Several other slight variations on this process are seen throughout the literature. Although this methodology of edge detection and line detection is most prevalent, there are outlier examples that locate the rails in the image via other techniques. One such method is also seen in [4], where detected lines are found using a Gabor filter. Another unique rail detection method can be seen in [18], where a Histogram of Oriented Gradients (HOG) was used to analyze the image and a region growing algorithm is developed to detect the rails based on

this information. The outcome of this process can be seen in Fig. 2.5. Finally, there are a few instances of neural networks being utilized to detect or segment the rail regions themselves. This process is similar to some utilized in region detection as mentioned above, but instead of training a system to identify the entire track region, it is trained to identify only the rails themselves [27] and [8].

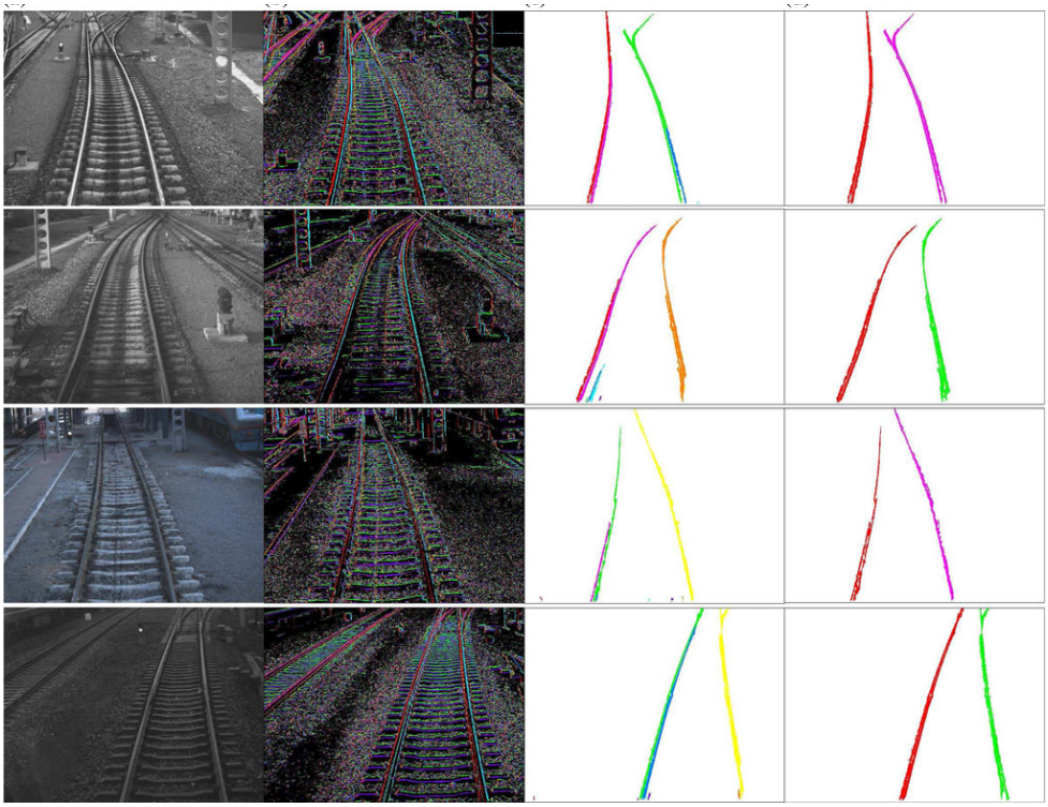


Figure 2.5 Example of unique rail detection method from [18] that utilizes a histogram of oriented gradients followed by a region growing algorithm.

The process of detecting a railroad by detecting the rails is possibly the most prevalent in the literature and this is for good reason. It isolates the most consistent and identifiable visual feature of a railroad which allows for a highly robust system. Also, the rails provide the most useful information for the next step in the process, namely guiding the UAV to follow the rails. Finally, most of the proposed line detection methods are only utilizing basic computer vision techniques allowing

the computations to be relatively lightweight. This is quite important if processing is to be done onboard the UAV. There are, however, some drawbacks to this technique. Firstly, a problem arises due to the fact that this method primarily relies on the rails being the only long, straight lines in the image. This is true for most cases when the railroad is in the wilderness, but if the railroad passes into an urban environment with many more straight lines the system can begin to make false identifications. Also, the simplicity of these techniques can be their downfall. Assuming a line detection algorithm accurately identifies rail lines in an image, its output will be a series of endpoints correlating to the identified lines. This output is the most useful information for following a railroad, but it is not the only useful information that can be attained. In contrast, a region detection approach retains all of the pixels in the full track region. This provides information on the railroad ties, the track area, and often the space immediately surrounding the track. This information is less useful for the purpose of navigation, but it can be used for other steps in the process such as obstacle detection, rail condition analysis, defect detection, etc.

2.1.3 Compound Region-Rail Detection

Recognizing the strengths of both region detection and rail detection, there are a couple examples of researchers advocating for a combined method that first applies region detection to locate the general track region accurately and subsequently employs rail detection within this region for finer localization. For instance, in [22] a simple HSV (Hue, Saturation, and Value) color extraction is performed, as explained above, to isolate the track region. After this is identified, line detection can be performed only in the track region to identify the rails themselves, isolating the most useful information for navigation. To achieve this the researchers use canny edge detection followed by a nearest neighbor algorithm for line detection. The greatest benefit of this compound method is in mitigating the major issue with the line de-

tection method. By first isolating the region of the track, one drastically increases the likelihood of the rails being the only long, connected lines. This decreases the possibility of falsely identifying lines from buildings or power lines because only the region in which the track is located is searched. This compound method also allows the excess information obtained by region detection to be retained in the system. The system can then utilize this information later to perform various means of track inspection.

The researchers in [23] have also developed a system that utilizes a compound method of region detection and line detection. To detect the track region the researchers develop a rail tie color matching algorithm. For this, a sample rail tie image is used as the ground truth. The image is then scanned to identify regions with similar pixel values to the ground truth, identifying all of the rail ties. This entire detected region is labeled as the track. The second stage of the system is to detect the rail lines within the track region. To accomplish this, the researches utilize the Hough transform line detection method. As is suggested above, this method provides the information required to develop a control system for following the track and retains the region information for further use.

This compound approach seems the most promising in a robust track detection algorithm, particularly when considering the broader goal of autonomous track inspection. Unfortunately, however, this approach is scarcely present in the literature, being utilized in only the two examples above [22] [23]. Even in these implementations, the processes utilized for region detection do not represent the most robust options as outlined above. This could be due to the increased computational load for techniques such as segmentation networks, but recent developments in such networks allow for this possibility [20]. For this reason the system outlined in Chapter 3 utilizes a compound region-rail detection approach which can produce sucessful track detection such as in Fig. 2.6.

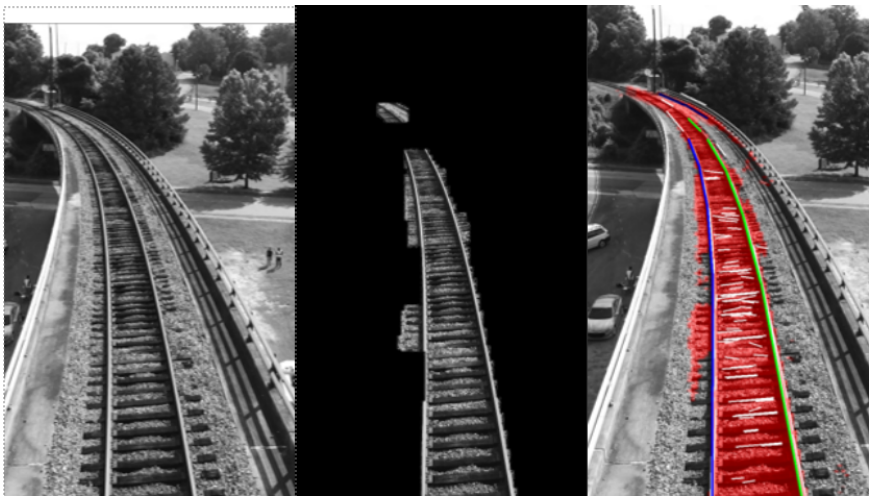


Figure 2.6 Outline of TrackNet process. Original image on left, detected track region in center, and rail detection on right. [13]

2.2 Track Following

Once the track has been identified, this information must be translated into a form useful for UAV controls. Fundamentally, four quantities are required for this: the current state of the UAV, the desired state of the UAV, the difference (or error) between these two states, and the change required to decrease this error. In the literature, there have been essentially two methods of quantifying this information, here called vanishing point detection and trajectory detection.

2.2.1 Vanishing Point Detection

Vanishing point detection assumes the input camera is facing forward, along the direction of the track. From this perspective, the rail lines will visually converge as two lines approaching each other. If these lines are extended to intersect, the point of intersection is called the vanishing point, which is assumed to approximate the direction that the UAV should be heading, thus representing the desired state of the UAV. It is then assumed that the center column of the camera's frame indicates the current heading of the UAV, providing the current state. The horizontal distance

between this and the vanishing point represents the state error. A helpful figure for visualizing this technique is in [17] and can be seen in Fig. 2.7. Finally, by determining which side of the frame's center column the vanishing point is located, the change in yaw that the UAV must undergo to decrease this error is easily identified. A system that controls this error to zero while the UAV is maintaining a constant forward velocity should guide the UAV to follow the track.

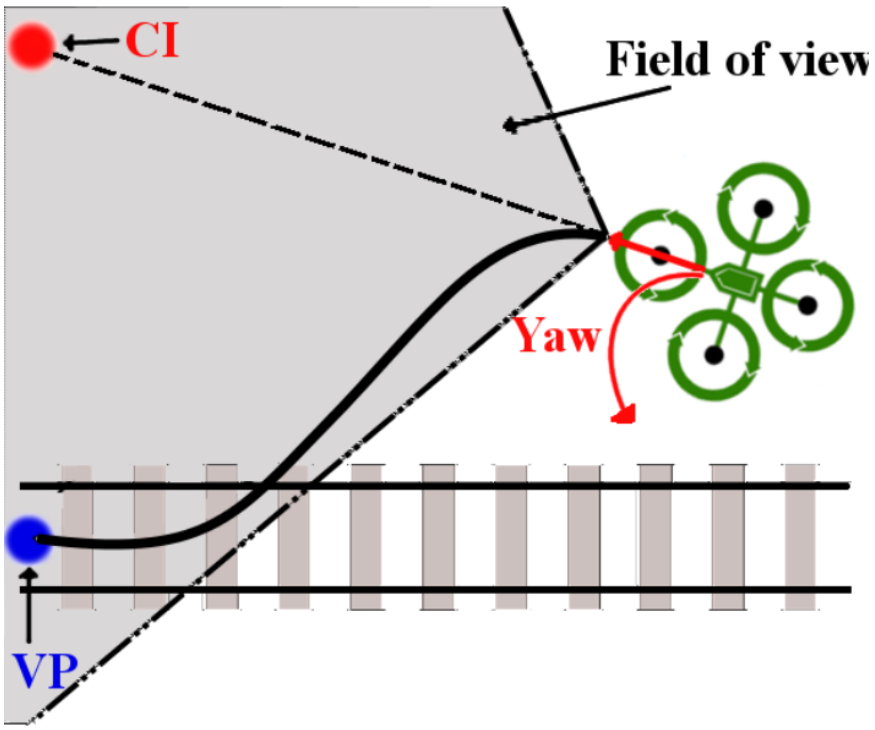


Figure 2.7 Diagram describing vanishing point technique from [17]. Shows the method of moving the center of the image (CI) to the vanishing point (VP) and how this will allow the UAV to follow the rail.

For example, in [9], the rail lines are detected using first canny edge detection and then the Hough transform. The researchers then argue that "the lines obtained as a result of the Hough transformation applied on the rails image will intersect at the vanishing point" [9]. A representation of this result from the UAV's perspective can be seen in Fig. 2.8. These researchers however encounter a common problem with

this approach, namely that if several predictions of the rail lines are made, a decision arises in determining which intersection of which lines is the true vanishing point. These researchers choose to solve this problem by collecting all of the intersection points and utilize the least squares solution to determine choose a vanishing point. This method proves successful in their testing on straight sections of track. The outcome is however less certain of accuracy when encountering curved sections of track as the lines will point in different direction depending on how far along the curve they are located. This issue is faced in Chapter 4 and another solution is justified and implemented.

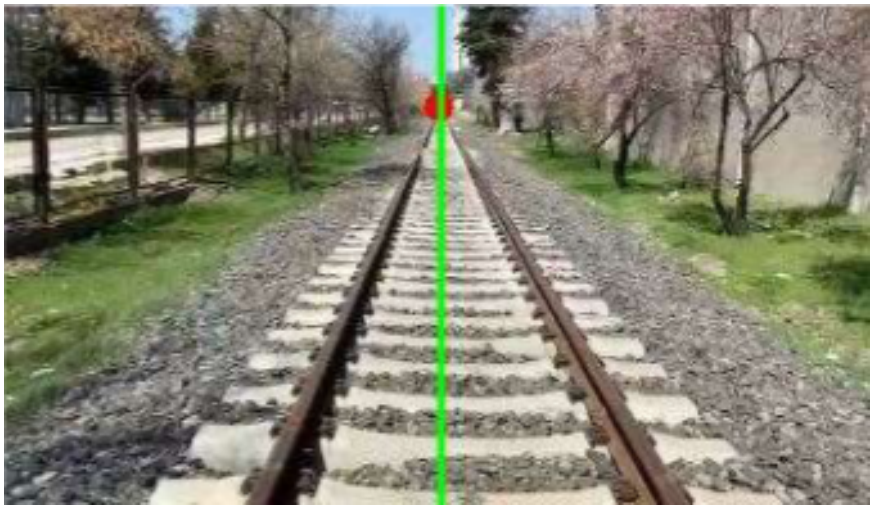


Figure 2.8 Example of detected vanishing point (red dot) in line with the midpoint of the screen (yellow line).

The researchers in [17] also implement a similar vanishing point detection system. Again, a line detection system is presented utilizing Laplace edge detection and Hough line detection. From these lines the researches suppose "the most dense area of crossing points of these lines will be the vanishing point of the tracks" [17]. To track the vanishing point, a Kalman estimator is developed using a simplified constant velocity, random walk model to describe the behavior of the vanishing point. This allows the system to maintain the best estimate of the vanishing point as well

as predict past, present, and future states.

Now that the vanishing point has been detected, all required quantities are available and a control system must be developed to "maintain at zero the horizontal distance between the vanishing point and the center of the image" [17]. This horizontal distance is therefore considered to be the state error and the control system seeks to keep this at zero by changing the direction of the UAV. In the literature the associated controls tend to assume a constant forward velocity and gives control effort in the yaw or roll channels. Both [17] and [9] utilize a PID controller to control the yaw channel. An example of this implementation is in [17] where the system is tested in a simulated environment as seen in Fig. 2.9.

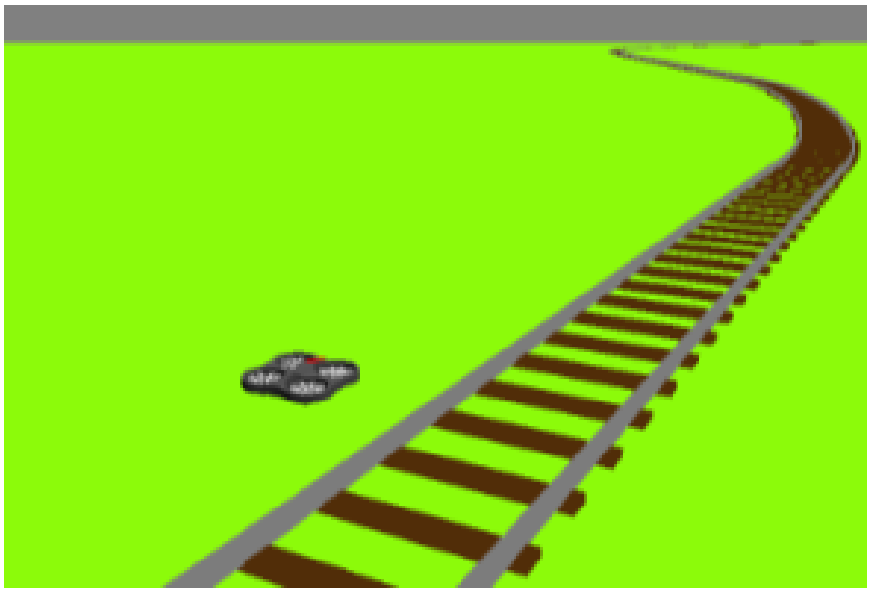


Figure 2.9 Example of vanishing point approach being implemented in a simulated environment in [17].

The vanishing point approach is a straightforward and effective method. It utilizes the nature of railroad lines and perspective to quickly and easily estimate the direction that the track is headed. It does not require any complex calculation or external information besides the rail lines calculated previously. Also, this approach is directly linked to line estimation, so if further development is done to account for

track intersections, for instance, the entirety of the adjustment can be performed in rail detection and will not cause any issues in the controls themselves. Due to perspective this approach does, however, require the usage of a forward facing camera for a vanishing point to visually exist. Altogether, the vanishing point approach to track following is a highly efficient and simple method, but it lacks in precision and robustness.

2.2.2 Trajectory Detection

The other interpretation technique seen in the literature is trajectory detection. In contrast to the vanishing point approach, this technique assumes a downward-facing camera and aims to keep the UAV directly above the tracks, while maintaining a constant forward velocity. To achieve this, the center line of the track, referred to here as the trajectory, is identified and considered the desired state of the UAV. An assumption is then made that the center point of the video frame can be estimated as the UAV's position projected onto the plane of the track. That point can then be designated as the UAV's current state, with its distance from the trajectory representing the state error. A control system can then be developed to turn the UAV in a way that decreases this error.

The calculation of railroad trajectory from camera input is essentially the process of identifying a curve that follows the center of the railroad track. This can be done based on a track region detection or a rail line detection. The researchers in [23] provide an example of both of these options. First, they seek to identify and follow roads by detecting the road region and calculating a trajectory from that. Once the region has been identified, they arrive at the same state that a track region detection system would find itself. The researchers then skeletonize the region's segmentation mask and apply a polynomial fitting algorithm to identify the remaining center line of the region. This process is shown to be effective for road following and an example

of this process is seen in Fig. 2.10. Additionally in [23], the researchers calculates trajectory based on rail line detection. This approach is much simpler as the mid-line between the two rail lines is assumed to be the center line of the track, or the trajectory. The data utilized in this method, however, seems to only contain straight sections of track where both rails are clearly visible. The simplicity of this trajectory calculation may encounter issues if any of these conditions were not met. Such issues are addressed in the system developed in Chapter 4.

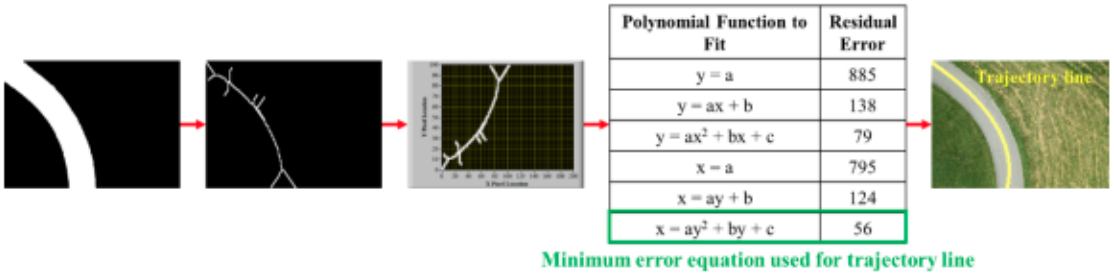


Figure 2.10 Example of trajectory detection based on region detection from [23]. Here the region of a road is masked, skeletonized and approximated as a trajectory line.

Once the desired trajectory has been identified, the relative location of the drone must be estimated. In [23] the same assumption is made as above, so the current state of the UAV is estimated as the center point of the frame. The distance between the center of the image and the desired trajectory is then considered to be the state error and can be controlled to be zero.

Another method that utilizes line detection to calculate drone trajectory is in [3]. Although this system is seeking to follow water channels instead of railroads the same basic principle is present. Once lines are detected, the problem becomes the same as with railroad following. The process outlined in [3] is simple and effective: detect the desired lines, determine the average direction of the lines, then use the difference between this direction and the vertical axis of the image as the error in trajectory. This

approach assumes the center column of the video frame is a projected velocity vector on the ground plane, therefore the current heading of the UAV (current state) can be approximated by this line. The desired state can then be estimated as the average orientation of all detected lines. The primary benefit of this process is due to the fact that the system averages together a larger pool of lines in trajectory calculation, helping to decrease the influence of false positives in trajectory calculation. On the other hand, this simple orientation control may not be sufficient on a live implementation. The main difficulty is caused by the trajectory calculation having no reference to the location of the track within the frame. An example of this error could be seen if the detected lines are in frame, but not centered. The algorithm would determine the orientation, but have no way of correcting this positional error and it may continue to increase in the noise of live situations. For this, some estimation of the detected line's position within the frame as in [23] would produce great improvement. For this reason, the system we developed performs control of two degrees of freedom, namely the yaw and roll. This essentially performs the orientation control as in [3] as well as horizontal position control as in [23]. This is further detailed in Chapter 4.

As a whole, the trajectory approach has many unique features and with the assumptions made it is highly robust and capable. It provides not only an estimate of the drone's location relative to the rail, but also the desired path for whatever length of rail can fit within the image. This allows for more comprehensive control and path planning techniques. However, in contrast to the vanishing point approach, this technique requires a downward facing camera. This, by nature, creates some limitations and also causes problems in calculations. A downward facing camera has a limited area of view when compared to a forward facing camera. This means that features on the track must be closer to the UAV before they enter the camera's frame. Therefore, the amount of time that the system will have to process any given section of track will be shorter potentially limiting the possible speed of the system. This also be coun-

teracted by flying the UAV higher, but his would require a higher definition camera thereby increasing processing time and cost. On the other hand, the downward-facing perspective makes it much easier to identify the location of obstructions relative to the track during inspection. It is much clearer what is actually obstructing the track and what is outside of its boundaries.

Altogether both approaches for rail following, the vanishing point approach and the trajectory approach, have advantages and disadvantages. The vanishing point approach is simpler and more computationally efficient while the trajectory approach is more accurate and robust. In the following chapters (Chapter 4, 5, and 6) these differences are tested and compared with implementations of both approaches.

Before such an implementation is created, however, track detection must be performed. In the next chapter (Chapter 3) a novel track detection approach named TrackNet is outlined. This approach begins with the methods seen in the literature, but integrates them in a novel way requiring additional developments. This system is an example of the scarcely represented compound method for track detection, beginning with region detection followed by rail detection. In contrast to the above compound methods, TrackNet utilizes a convolutional neural network for region detection that will prove to be much more robust than what is currently seen in the literature. Additionally, the rail detection includes a more sophisticated line selection algorithm that is not present in the literature. This extra step allows for significantly increased performance in identifying the rails when observing curved sections of track. With this improved track detection system, track following can commence and will be outlined in Chapter 4.

Chapter 3

Manuscript 1, "UAV-Based Railroad Line Detection"[1]

3.1 Abstract

The railroad industry is crucial for modern transportation, therefore the need for maintaining the integrity and safety of rail infrastructure is immense. Traditional rail inspection methods, involving manual teams or specialized vehicles, are labor intensive and costly, causing logistical inconveniences, especially for rapid, large-scale inspection. This paper explores and expands the adaptation of Unmanned Aerial Vehicles (UAVs) and advanced computer vision for rail inspection. While existing literature highlights the benefits and capabilities of UAVs, challenges persist, and a fully integrated, online system has yet to be thoroughly implemented and tested. We seek to create a system that performs the task of track following strictly by visual sensor perception, eliminating any reliance on GPS and ensuring autonomy in environments with limited or degraded GPS availability, such as dense settings, tunnels, etc. The system will perform all processing onboard, providing immediate results without the need for external processing or infrastructure. Our proposed approach divides the problem into track detection, track interpretation, and track following. This work focuses on the first of these steps, track detection. We survey existing approaches, assess their strengths and limitations, and introduce a novel

[1]Keith Lewandowski and Nikolaos Vitzilaios. "UAV-Based Railroad Line Detection". In: *Joint Rail Conference (JRC)*. 2024.

method addressing prior challenges, keeping in mind the goal of a fully integrated, autonomous system for rapid track assessment.

3.2 Introduction

In order to ensure sufficient safety, efficiency, and reliability in railroad networks, effective inspection is essential. Traditional methods of inspection accomplish this goal, but often have several drawbacks such as being time-consuming, costly, and prone to human error. A modern solution to this problem is the use of Unmanned Aerial Vehicles (UAVs), widely known as aerial drones; these drones can rapidly inspect large stretches of railway and require little to no preparation or infrastructure. In the aftermath of destructive events like hurricanes or landslides, drones can conduct fast and thorough inspections of large track sections at a moment's notice. If such an event is thought to have caused an obstruction, a locomotive can simply deploy a drone to search the track ahead and report if there are any major issues. Alternatively, if a large section of the track is thought to be potentially problematic, one or several drones could be deployed to locate the problematic areas and report back. In any such case, no preparation or pre-existing infrastructure is needed. Also, there is no need to alter the normal functions of the tracks or bring in additional workers to enable inspection. A major component in enabling the development of such a system is the use of computer vision and machine learning techniques to enhance the UAV's capabilities for automated track inspection. Drones capture high-resolution images and videos in which computer vision algorithms automatically identify track features. Additionally, the use of computer vision techniques for navigation, creates increased robustness and independence when compared to other path planning approaches. The drone's camera and processing are controlled onboard, eliminating any dependance on external signals such as GPS as well as the need for manual input. The only requirement for a vision-based system to function is the presence of adequate light, which

can also be provided by additional onboard equipment. This allows for operation in GPS degraded/denied environments such as dense urban areas, remote rural areas, or inside structures such as tunnels. This approach also automates path planning, removing the need for manually created flight paths. Overall, the use of drones in railroad inspection is not only an effective and efficient solution, but also fills gaps in current, traditional methods of inspection. For these types of systems, computer vision techniques are robust in achieving autonomy and require little dependance on external and less-reliable infrastructure. This paper will explore specific applications of computer vision techniques for railroad inspection, present a fully developed track detection system for this purpose, deemed TrackNet, and lay the foundations for a fully integrated drone system for online track inspection.

3.3 Background

This section outlines diverse methods for identifying railroad tracks in images or videos as found in the literature. Approaches vary based on the railroad features that are detected and used for identification. Some methods focus on recognizing the entire track region by identifying common visual features. Others prioritize the most consistent visual feature of railroads, the rails themselves. Additionally, certain approaches utilize both methods together, first identifying the track region and then detecting the rails within it. The subsequent sections will delve into these categories, exploring common techniques in each for effective railroad track identification.

3.3.1 Region Detection

The first method of railway detection involves identifying a full region within an image containing the railroad tracks. This approach typically begins with image preprocessing methods which can enhance track features and reduce noise. These techniques include alterations like converting the image from RGB (red, green, blue)

space to HSV (hue, saturation, value) space, adjusting brightness, thresholding, or blurring. Some studies, such as [22], pinpoint specific HSV values common to railroad regions to enable the identification of such regions. Converting an image into HSV space, simply describes the same image in a different way, namely by determining each pixel's hue, saturation, and value. These researchers recognize that when observing a rail image in this way, the pixels that constitute the track region typically fall within a certain range of hue values. Fig. 3.1 (bottom image) displays the hue representation of one such image where the track shares a common value. When this value is isolated, it results in the top image in Fig. 3.1, a near perfect detection of the track region. While efficient, these methods may lack robustness as they rely on features not universally consistent for all tracks in all environments. To address this issue, other approaches employ machine learning algorithms for rail segmentation [24]. These systems are typically trained on a large set of annotated data to become capable of detecting track regions in new images. The annotated data would be an image with a track and a ground truth mask (silhouette-like outline) of the correct track region. With enough data like this, the system can determine the deep features in an image that are most likely to constitute a track. Then, if properly trained on a suitable dataset, the system can use the same features to predict the region most likely to be a track in a new image. In [26], we see another example, where a semantic segmentation algorithm is applied to identify track regions. Region detection proves advantageous in railroad inspection, isolating the entire track area for comprehensive analysis. This enables inspection techniques such as obstacle detection by also providing the important regions to be searched. This method provides a useful measure by which the drone can navigate as well as a necessary component for the inspection stage.

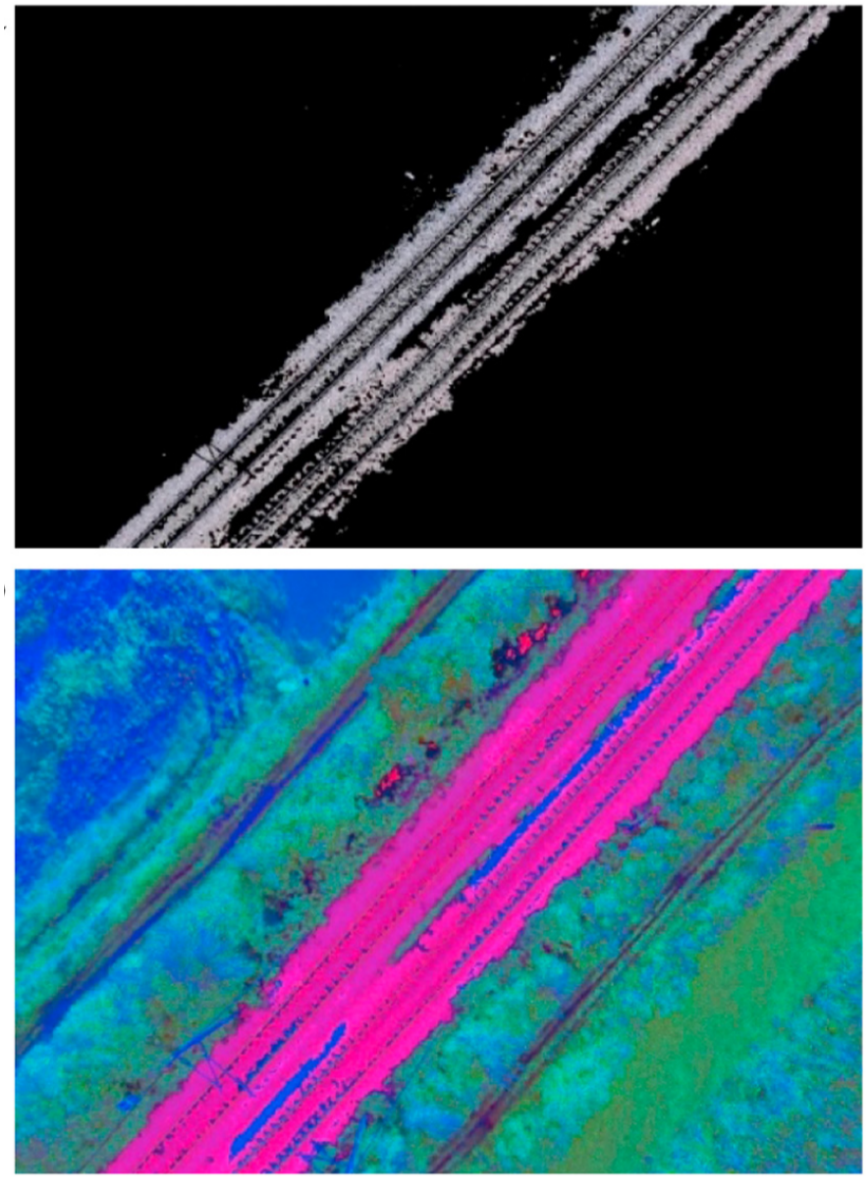
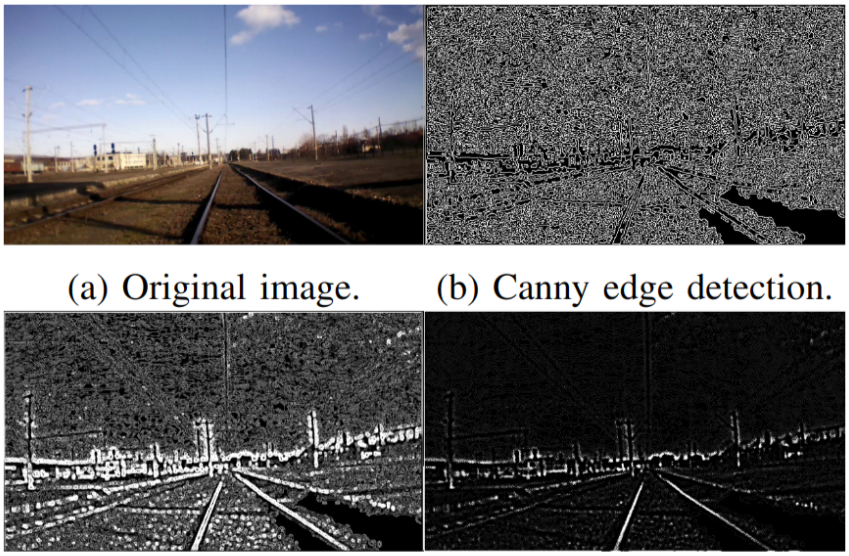


Figure 3.1 Region detection through HSV value isolation [22]

3.3.2 Rail Detection

The second computer vision method for rail detection focuses on identifying a specific feature of the rail image: the track rails. Instead of detecting the entire track region, this method isolates the visually distinct rails. Recognizing the long, distinct lines of the rails allows for determining their position and, consequently, the track's direction. Of the various approaches that have been explored for this purpose the most common



(a) Original image. (b) Canny edge detection.

(c) Sobel edge detection. (d) Laplacian filtering.

Figure 3.2 Comparison of edge detection algorithms [4]

is the combination of edge detection and line detection, as seen in [16], [2], [17], [11], [4]. Edge detection, a foundational technique in computer vision, identifies abrupt changes in image intensity or color using methods like Laplacian, Canny, and Sobel algorithms. One comparison of these three methods can be seen in Fig. 3.2, where the three methods are applied to the same image with mixed results. Based on this example, it is clear that Sobel performs the best, however, in the literature, preferences between these three techniques vary; [4] favors Sobel, [16] opts for Canny, and [17] recommends Laplacian. Usually, these algorithms are complemented by image preprocessing techniques such as blurring, thinning, and thresholding to optimize edge detection, tailored to specific approaches for enhanced results. This also explains the different technique preferences because results can greatly depend on the processing steps prior to edge detection as well as the needs of the following steps in the system. After detecting edges in an image, the algorithm faces the task of discerning which edges represent rails. Given that rails typically manifest visually as long, continuous lines, line detection becomes a natural choice for this step. The widely employed line

detection method in the literature is some form of the Hough Transform, typically the Probabilistic Hough Transform. In depth explanation and the history of these algorithms is thoroughly explained in [10]. Various preprocessing techniques, such as blurring, thinning, and thresholding, enhance the image before line detection, similar to the edge detection process. After identifying lines, a common post-processing step involves filtering the detected lines to pinpoint the lines most likely to be rails. For instance, some researchers eliminate nearly horizontal or vertical lines, reasoning they are unlikely to be rails in certain cases. While edge and line detection dominate, outliers explore alternative techniques. Some use a Gabor filter which attempts to isolate lines at particular angles in the image ([4], [9]). If the orientation of the track within the image is known, there is a likely range of angles that the rails lines will fall into. These researchers utilize this to estimate where the rail lines are located and, more importantly, where they are heading. Another approach in [18] utilizes a Histogram of Oriented Gradients (HOG) to attempt to map gradients of pixel values within the image as well as the orientations of these gradients. A representation of this process can be seen in the left two columns of Fig. 3.3. Due to their continuous color, it is assumed that the rail lines will have a nearly non-existent gradient along their length. Because of this, the researchers use a region-growing algorithm to begin from the bottom of the image and follow these rail gradients up the image. The result is an identification of the rail lines as can be seen in the right two columns of Fig. 3.3. Finally, neural networks are another method employed for rail identification. Here, a network is trained to recognize and often mask the region of the rails themselves as in [8] and [27]. This rail-centric approach is favored for its ability to isolate the most distinctive visual feature, the rails, guiding UAVs effectively with lightweight computations. However, challenges arise in urban environments where assumptions about rails being the only long, straight lines break down. Additionally, the simplicity of this approach aids computational load but sacrifices useful information. To strike a

balance, some researchers opt for a hybrid of region and rail detection, combining the strengths of both methodologies.

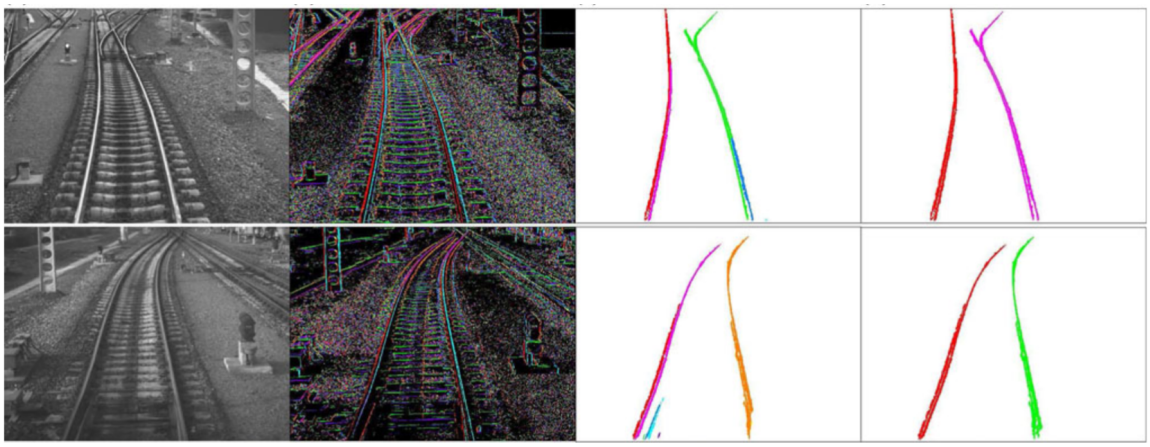


Figure 3.3 Rail detection using HOG and region-growing algorithm [18]

3.3.3 Compound Methods

A compound method combines track region detection with subsequent rail detection within the identified track region. In [22], HSV color extraction isolates the track region, and rail detection, using Canny edge detection and a nearest neighbor algorithm, identifies the rails for navigation. This type of approach mitigates issues with rail detection by narrowing the search to the track region, reducing the likelihood of false identifications from other structures. Additionally, it retains excess information obtained during region detection for potential track inspection later in the process. In [23], researchers introduced a compound method involving region and rail detection. They employed a rail tie color matching algorithm to identify the track region, using a sample rail tie image as the ground truth. The Hough transform line detection method was then applied to detect rail lines within the track region. This approach not only provides the necessary information for a control system to follow the track but also retains the region details for future use. The result of this method can be seen in Fig. 3.4. The boxed in rail tie was used as a sample to scan

the image for matching color signatures. From this the track region was identified as shown in the larger box around the track. This region was then searched for lines which were identified and drawn over the rails in the image. It's clear that various methodologies for identifying railroad tracks have been explored in the literature; preliminary research has been done in all areas, region detection, rail detection and compound methods. Region detection, focusing on the entire track area, proves advantageous for preserving valuable information and facilitating subsequent processes. Rail detection, targeting the distinct visual feature of rails, is widely favored for its simplicity and effectiveness, though it may encounter challenges due to changing and difficult environments. Compound methods, combining region and rail detection, offer a balanced approach, mitigating issues and retaining helpful information. These diverse techniques contribute to the development of robust systems for railway track identification in images or videos.

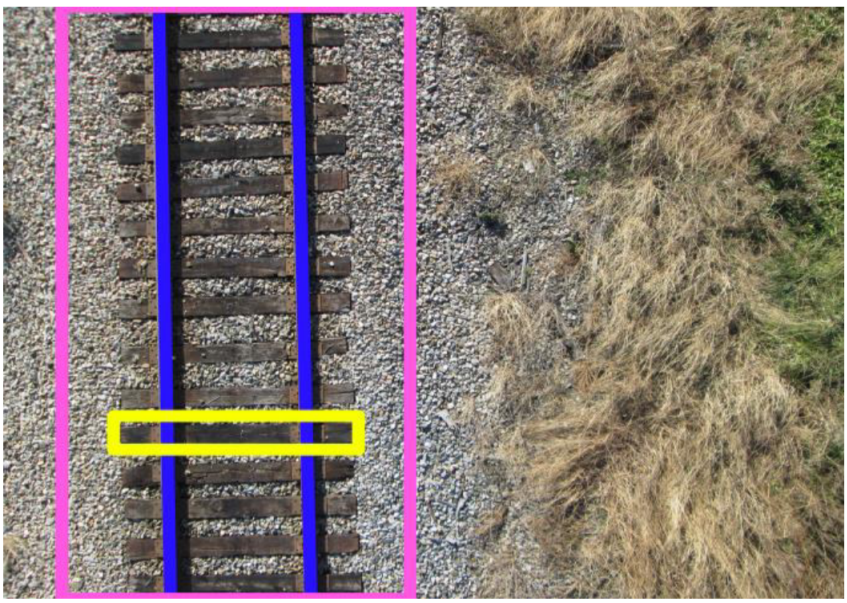


Figure 3.4 Compound region-line detection [23]

3.4 Methodology

As mentioned above, the system proposed in this paper focuses on the first step of drone-based track inspection, track detection. This system seeks to have minimal sensor reliance, employing only an onboard camera. This approach, in spite of its many benefits, requires powerful computing for complex algorithms. The system must quickly and effectively interpret visual information from the onboard camera to determine the drone's current position and make navigational decisions. As detailed in the literature review, two primary approaches for track identification, region detection and rail detection, were considered. Region detection involves a more comprehensive identification of the entire track area, including the rails, the ties, the surrounding land, etc. On the other hand, rail detection targets a specific feature within this region, namely the rail lines. The literature underscores that region detection is a powerful, yet costly method. Often utilizing machine learning techniques, it can be quite computationally heavy when properly implemented. Conversely, rail detection is often distilled to a few simple and efficient processes, greatly decreasing its required computational load. Despite its computational efficiency, however, rail detection faces challenges in accurately selecting rail lines, particularly in complex scenarios and environments. This paper introduces a compound method, deemed TrackNet, integrating both track region detection and rail line detection. This strategic amalgamation capitalizes on the reliability of region detection and its utility in track inspection, while leveraging the efficiency of rail line detection. The compound approach aims to mitigate the drawbacks of individual methods and achieve a more balanced and effective solution, addressing the complexities of track detection and following. While certain compound methods resembling this approach have been explored in the literature ([23], [22]), some techniques in these implementations are rudimentary and depend on assumptions that are not universally applicable and/or involve some form of manual input, as outlined in the background section above. The

primary contribution of this paper is a fully autonomous system that will receive any image or video input and provide rail detection and associated flight commands. This is achieved by utilizing the state-of-the-art techniques for each stage of detection. The subsequent sections delve into TrackNet's comprehensive details, providing a robust foundation for efficient and reliable online track detection and following procedures.

3.4.1 Region Detection

The initial step in the TrackNet system employs a segmentation network, referred to as Unet in the referenced work [21]. Unet is a convolutional neural network designed for semantic segmentation, providing masks of identified regions instead of just class labels or bounding boxes. It assigns a class label and a confidence score to each pixel in the image based on how likely it is determined to be one of a set of classes. The accumulation of these classified pixels creates regions of undefined shape for the identified classes—in this case, 'track' or 'not track'. This approach is particularly advantageous for railroad tracks, given their diverse orientations and curvatures that don't conform to defined shapes suitable for bounding boxes. During the training phase of the Unet system, input is provided as a set of annotated data. This dataset consists of images containing railroads and the corresponding annotations of track regions. These annotations label each pixel as 1 or 0, indicating 'track' or 'no track.' Like other machine learning techniques, the system undergoes training on this dataset. The Unet network learns from this annotated data, generating weights for the detection phase. With appropriately tuned weights, the system can then process new images, producing a labeled mask akin to the training annotations. In this mask each pixel has a grayscale value indicating the system's confidence of the pixel being part of a track. Overlaying this mask on the original image completes the track region detection. An example of this can be seen in Fig. 3.5. In this system, grayscale conversion of images is performed to better suit Unet and mitigate the impact of lighting

and environmental variations on detection outcomes. The left image in Fig. 3.5 is an example of this. After generating the mask, a threshold is applied to isolate only the 'track'-labeled pixels with the highest confidences. All other pixels are discarded, and the remaining track region is all that is left as is clear in the left image in Fig. 3.5.

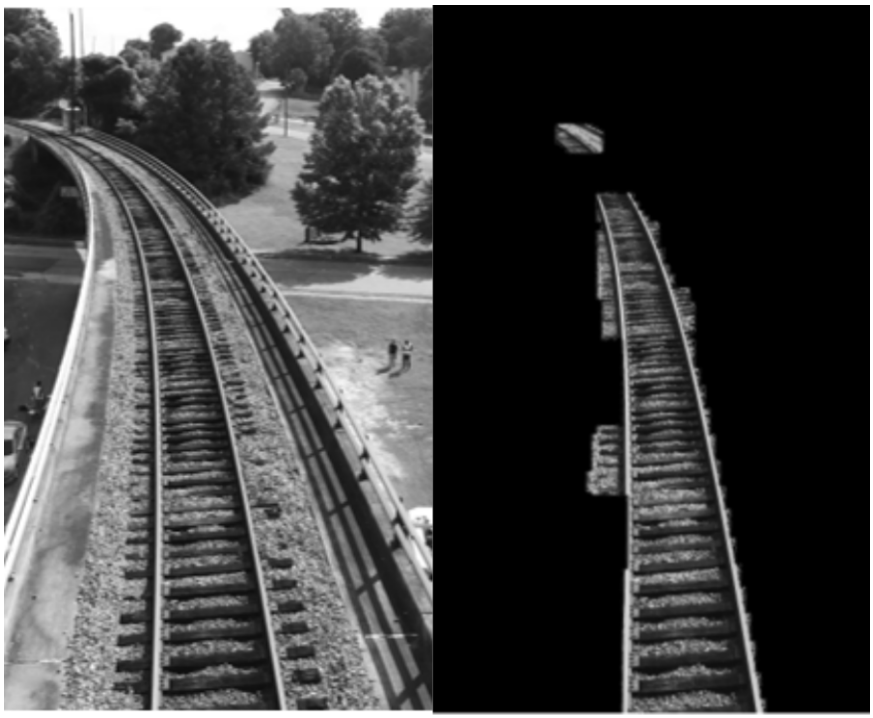


Figure 3.5 Step 1 of TrackNet: track region detection using trained Unet network

3.4.2 Rail Detection

Once the track region has been detected, the system will move to the rail detection stage. This stage utilizes similar techniques to those outlined in the related works above. The main goal of this step is to search within the track region, as detected in the previous step, for the rails. For this process it is assumed that the longest connected lines within the track region are the rail lines. The identification of these lines is done in three steps: edge detection, line detection, and line selection. When

searching for rail lines, performing line detection on the track region can yield numerous lines, some of which are unhelpful. As seen in the related works, edge detection can be applied to help solve this problem. This approach improves the efficacy of line detection by emphasizing areas with meaningful lines, namely edges. Common algorithms for edge detection include Canny, Laplace, and Sobel. As demonstrated later in this paper, a comparative analysis revealed that Canny edge detection performed best for the presented system. In this system, Canny edge detection is applied to a grayscale version of the original image, followed by overlaying the region mask. This can be seen as a translucent red overlay in Fig. 3.7. This sequence prevents the identification of edges along the mask's edge, avoiding potential inaccuracies in this stage. Once the edges within the track region have been identified, a line detection algorithm is performed to identify lines based on these edges. As a part of the edge detection comparison in the results section, three-line detection functions were also compared, namely, the Line Segment Detector (LSD), the Fast Line Detector (FLD), and the classic Probabilistic Hough Transform. The comparison showed that FLD following Canny edge detection was determined to perform best in the presented system. This function outputs a list of lines (characterized by two endpoints) within the image. Some of these lines will be the rail lines that are being searched for, but there will also be many additional lines that need to be eliminated. All such lines can be seen in Fig. 3.7 as white lines. Here it is clear that the lines in the track region include the rail lines, but also the railroad ties and other miscellaneous lines. The next task is to intelligently decide which of these lines are the rails. Once potential rail lines are identified, determining the actual rails involves more than just selecting the two longest lines. While sorting lines by length may work in some cases, it breaks down when rails are perceived as rectangles, potentially resulting in separate lines for each side of a rail. Issues also arise if the rail lines are not detected as continuous due to obstacles, curves, or algorithm errors. Therefore,

a more robust selection method is needed. In the literature, solutions such as filtering by angles or line length has been proposed. Considering these, a new robust solution is developed which includes eliminating duplicate lines, forming chains of lines, and then selecting the two longest elements. Each step of this process is done by a sequential comparison of every possible pair of lines detected previously. This comparison is done using the calculation of three parameters, namely endpoint distance, angle difference, and perpendicular distance. In these calculations, each line is defined by two endpoints, so each pair of lines, i, j, will include four endpoints, $p_{i,1} = (x_{i,1}, y_{i,1}), p_{i,2} = (x_{i,2}, y_{i,2}), p_{j,1} = (x_{j,1}, y_{j,1}), p_{j,2} = (x_{j,2}, y_{j,2}))$. It then follows that the simplest parameter is the distance between each endpoint calculated using the distance formula:

$$d_{ij,1,1} = \sqrt{(x_{i,1} - x_{j,1})^2 + (y_{i,1} - y_{j,1})^2} \tag{3.1}$$

$$d_{ij,2,2} = \sqrt{(x_{i,2} - x_{j,2})^2 + (y_{i,2} - y_{j,2})^2} \tag{3.2}$$

$$d_{ij,1,2} = \sqrt{(x_{i,1} - x_{j,2})^2 + (y_{i,1} - y_{j,2})^2} \tag{3.3}$$

$$d_{ij,2,1} = \sqrt{(x_{i,2} - x_{j,1})^2 + (y_{i,2} - y_{j,1})^2} \tag{3.4}$$

Where $d_{ij,1,1}$ is the endpoint distance between $p_{i,1}$ and $p_{j,1}$ and so on. The second parameter is the angle difference between the two lines, which is calculated using the $atan2$ function as:

$$a_{ij} = atan2(m_i - m_j, 1 + m_i * m_j) \tag{3.5}$$

Where m_i and m_j are the slope of line i and j respectively. The third parameter is the distance of each line's endpoint perpendicular to the other line and it is calculated as:

$$P_{ij,1} = \frac{(\vec{p_{i,2}} - \vec{p_{i,1}}) \times (\vec{p_{j,1}} - \vec{p_{i,1}})}{||\vec{p_{i,2}} - \vec{p_{i,1}}||} \tag{3.6}$$

$$P_{ij,2} = \frac{(\vec{p_{i,2}} - \vec{p_{i,1}}) \times (\vec{p_{j,2}} - \vec{p_{i,1}})}{||\vec{p_{i,2}} - \vec{p_{i,1}}||} \tag{3.7}$$

$$P_{ji,1} = \frac{(\vec{p_{j,2}} - \vec{p_{j,1}}) \times (\vec{p_{i,1}} - \vec{p_{j,1}})}{||\vec{p_{j,2}} - \vec{p_{j,1}}||} \tag{3.8}$$

$$P_{ji,2} = \frac{(\vec{p_{j,2}} - \vec{p_{j,1}}) \times (\vec{p_{i,2}} - \vec{p_{j,1}})}{||\vec{p_{j,2}} - \vec{p_{j,1}}||} \tag{3.9}$$

Where the notation $\vec{p}$ represents the vector of point p, $||\vec{p}||$ represents the vector norm of a vector $\vec{p}$, and $P_{ij,1}$ is the perpendicular distance of endpoint $p_{j,1}$ to line i.

With these parameters defined the steps of duplicate line elimination and line chaining are possible. Both of these processes are based on three conditions. The first two are simply that the angle difference and the perpendicular distances are small. If two lines are duplicates, they will have similar angles and their endpoints will have small perpendicular distances from each other. This, however, will also be true of two lines that are part of the same line chain, namely ones that may be along the same rail. If these two conditions are true, one final condition is needed to decide if the lines are duplicates or part of the same chain. This final condition is the number of endpoint distances that are small. This condition is used to determine the two line's proximities to one another. If none of the endpoint distances are small, the lines are not near each other at all and therefore are not duplicates nor are they part of the same chain. If one of the endpoint distances is small, the lines are likely end to end and should be chained together. If two of the endpoint distances are small, the lines are likely duplicates and the shorter one is removed. Lastly, if three or all of the endpoint distances are small, the lines must both be short and likely both need to be removed. This decision scheme is outlined in Fig. 3.6.

Once duplicates are eliminated and line chains are formed, the lengths of all elements (chains and individual lines) are compared, and the two longest exclusive elements are selected as the rails, completing the line selection process. In the example

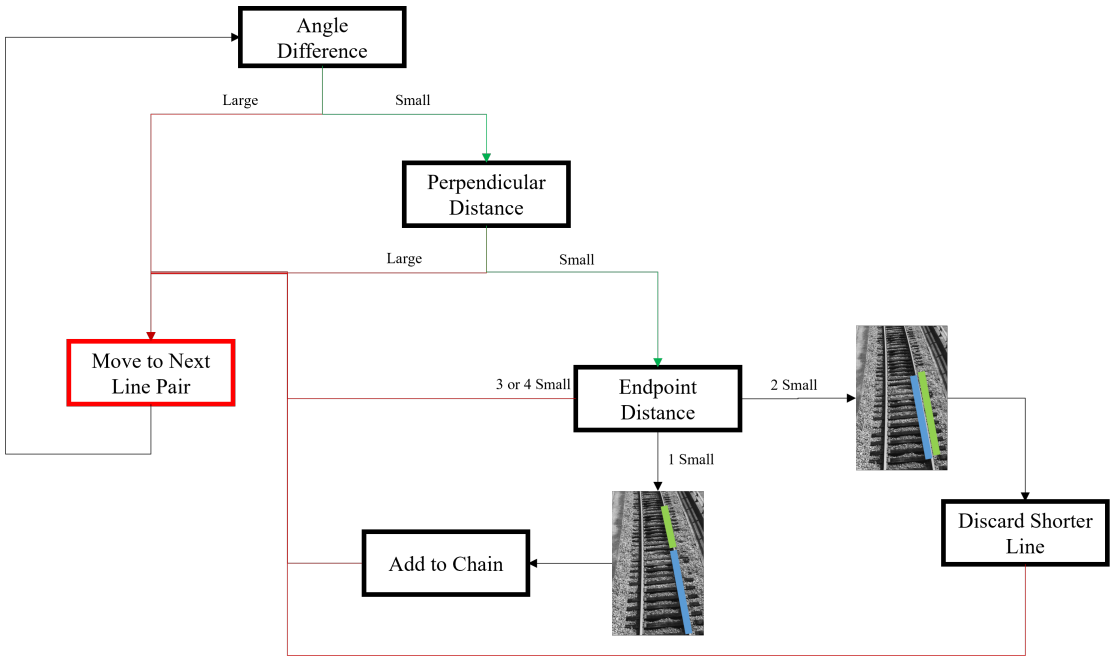


Figure 3.6 Outline of Line Selection Process

in Figure 6, the two elements identified as rails are highlighted in green and blue. Here the necessity of line chaining can be seen as the rail is turning and cannot be estimated as one straight line, but as chains of lines.

3.4.3 Track Interpretation

At this point in the process, track detection has been completed, having identified both the track region and rail lines in the image. While this information is valuable, it alone cannot provide input to a drone's control system. To enable this, a metric representing the relative position between the drone and the track needs to be calculated, estimating a type of error than can be controlled to zero. Two methods are proposed for this purpose: the vanishing point approach and the trajectory approach. Future testing is needed to determine their comparative effectiveness, so both are detailed here. Note that some information from the track detection stages may be unnecessary for one approach and redundant at times, these calculations

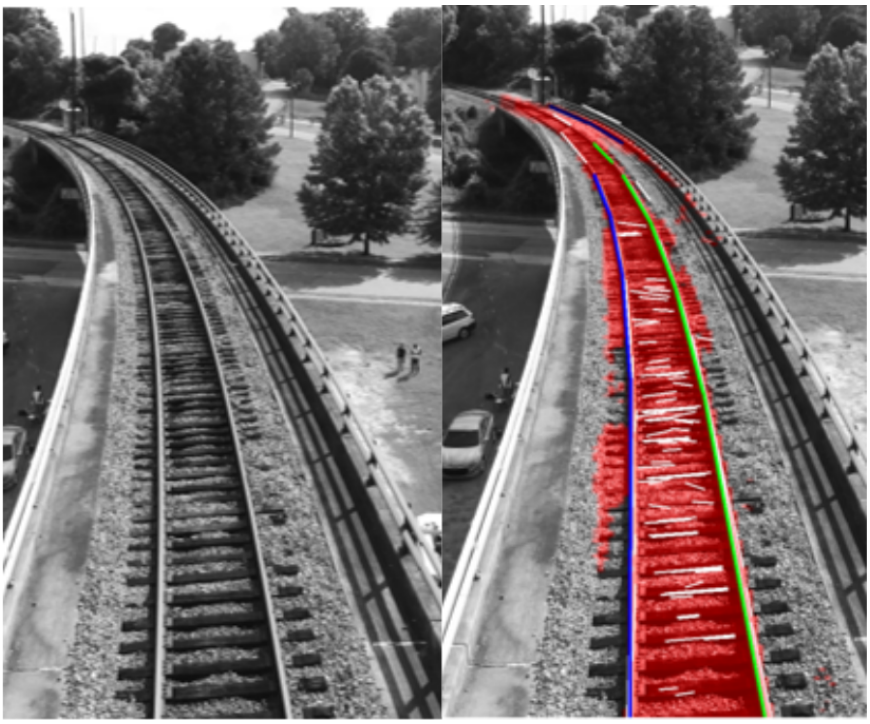


Figure 3.7 Step 2 of TrackNet: rail detection results using edge and line detection

would be omitted in a more specialized implementation. The vanishing point method leverages a characteristic of track images when the camera faces forward along the track. Due to perspective, the two rail lines appear to converge at a vanishing point. This point serves as an indicator of the track's direction. The drone's current state is considered to be the center column of the image, and the goal is to minimize the distance between the vanishing point and this column for effective track following. If the vanishing point shifts right, the drone turns right, and vice versa. The magnitude of the turn is proportional to the distance. The challenge lies in calculating the vanishing point, particularly when dealing with track curves. Tested methods include the intersection of the last lines in the chain, the average intersection of all lines, and the intersection of the longest lines. Comparative analysis favors the intersection of the longest lines for a useful vanishing point. In the lower image in Fig. 3.8, an example of vanishing point detection can be seen. The vanishing point is calculated using

the longest line in each element and is pictured as a green dot just left of the center line at the top of the image. Because this point is to the left of the center line the control indication written in the top left of the image is seen as "turn left" with an associated magnitude. In the case of a drone system with a downward-facing camera view, the desired drone position aligns with the center of the two rail lines. The drone's desired trajectory can therefore be estimated as the midpoint between these lines. This trajectory is the basis of the second track interpretation approach and is pictured in the top image in Fig. 3.8 as a white line. The drone's position relative to this trajectory is determined by assuming the center point of the image represents the drone's location when projected into the track plane, seen as a white dot in Fig. 3.8. If it can be assumed the camera angle is perpendicular to the track's plane, controlling the distance between the center point and the closest point on the trajectory (the black dot in Fig. 3.8) to zero ensures the drone follows the track. Considerations for comparing these approaches include camera orientation advantages and disadvantages. The forward-facing camera allows for a longer observed track length, aiding route prediction and faster drone speeds. In contrast, the downward-facing camera provides clearer scene depiction, facilitating improved obstacle and track identification. Assumptions for both approaches require validation through experimentation, emphasizing the need for testing and comparison in live implementations.

3.5 Results and Discussion

The primary contribution of this paper is not only a compilation and distillation of the state-of-the-art track detection and following techniques, but also a working implementation of the system. The showcased system, named TrackNet, represents a holistic development capable of processing input from images or video and generating drone control commands that can be adapted to any specific onboard drone implementation. The fundamental phases of this system encompass region detection,

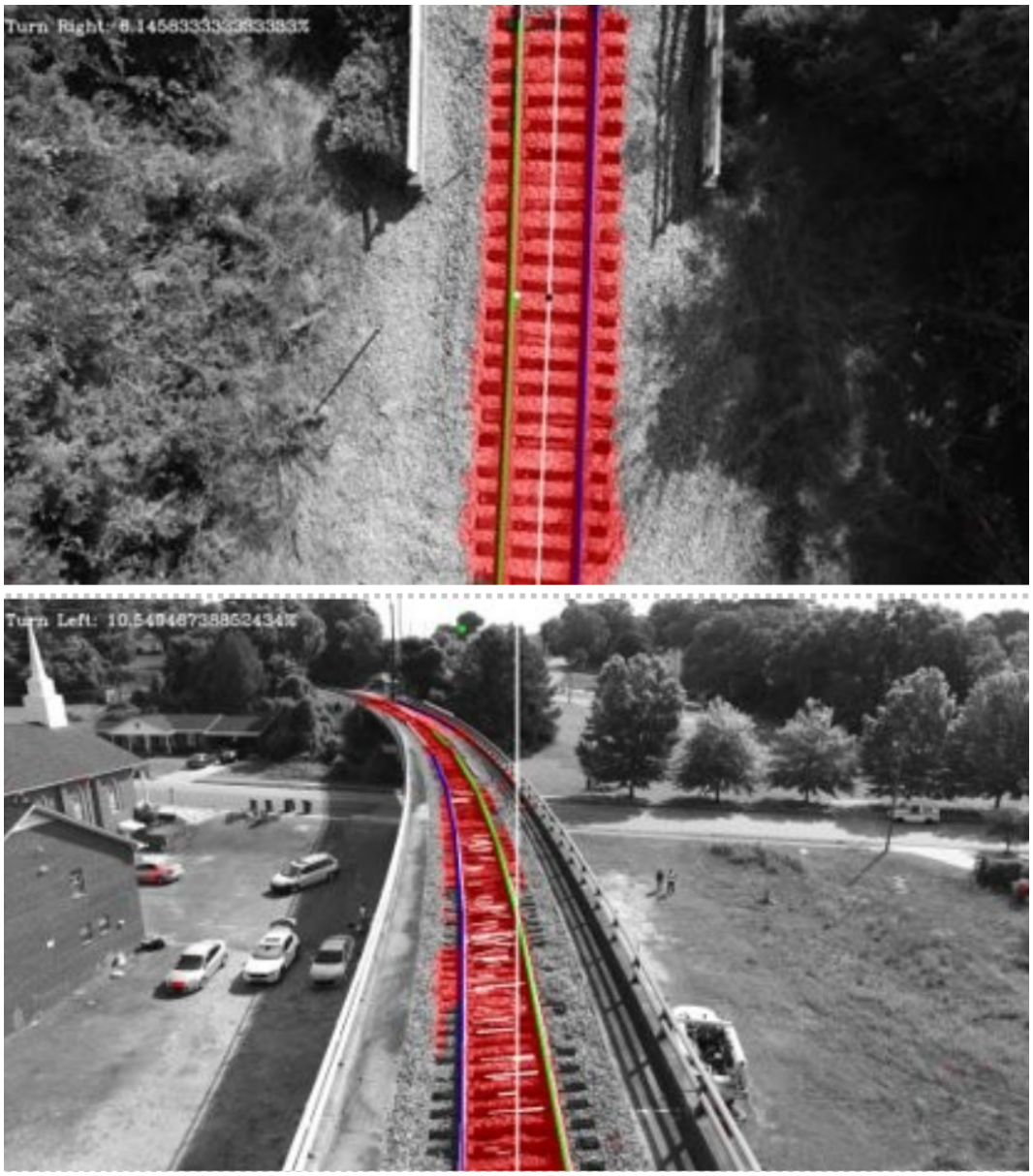


Figure 3.8 Example of trajectory calculation (top) and vanishing point calculation (bottom)

rail detection, and rail interpretation. The methodologies explained earlier were subjected to testing and comparison, paving the path toward a complete implementation for on-board deployment on a drone.

3.5.1 Region Detection

The Unet network incorporated in TrackNet underwent training using 7,293 images sourced from various freely available online platforms, including RailSem19 [28] and Google Earth. This dataset was comprised and annotated for training using the labelme free application [25]. During this training the accuracy and loss plots were constructed and can be seen in Fig. 3.9. In these plots one can see the network's accuracy and loss as it updates the weights after each epoch. In the accuracy plot (top) it is clear that the network improves quickly in the beginning and then has a slight incline to the end. Most of the major problems are sorted out quickly and only minor adjustments are made in the later epochs. The loss plot is similar, but opposite. Loss is a measure of the error of the system and is applied to its detection on the training dataset as well as the testing dataset. The more important metric here is the test line as the system is optimizing mostly to decrease this loss, so that it will be most capable of functioning on new data, not that which it was trained on. Upon scrutinizing the outcomes, it is evident that the system excels in discerning track regions, both in downward and forward-facing perspectives. Challenges are noted in close-up images where the track region dominates the majority of the frame. Additionally, the system demonstrates room for improvement when confronted with structures sharing similar features with tracks, such as power lines and bridges. Many of these challenges can be addressed by refining the training dataset to encompass more diverse scenarios and incorporating additional post-processing steps. Despite these minor shortcomings, the region detection component of the system is deemed more than satisfactory for the envisioned drone system, with potential enhancements achievable through subsequent post-processing. Some example results can be seen in Fig. 3.10. Here, the left column is the grayscale image input into Unet and the right column is the detection result. The system is seen to be successful on simple, straight lines (top row), more complex, small, curved lines (bottom row) as well as situations

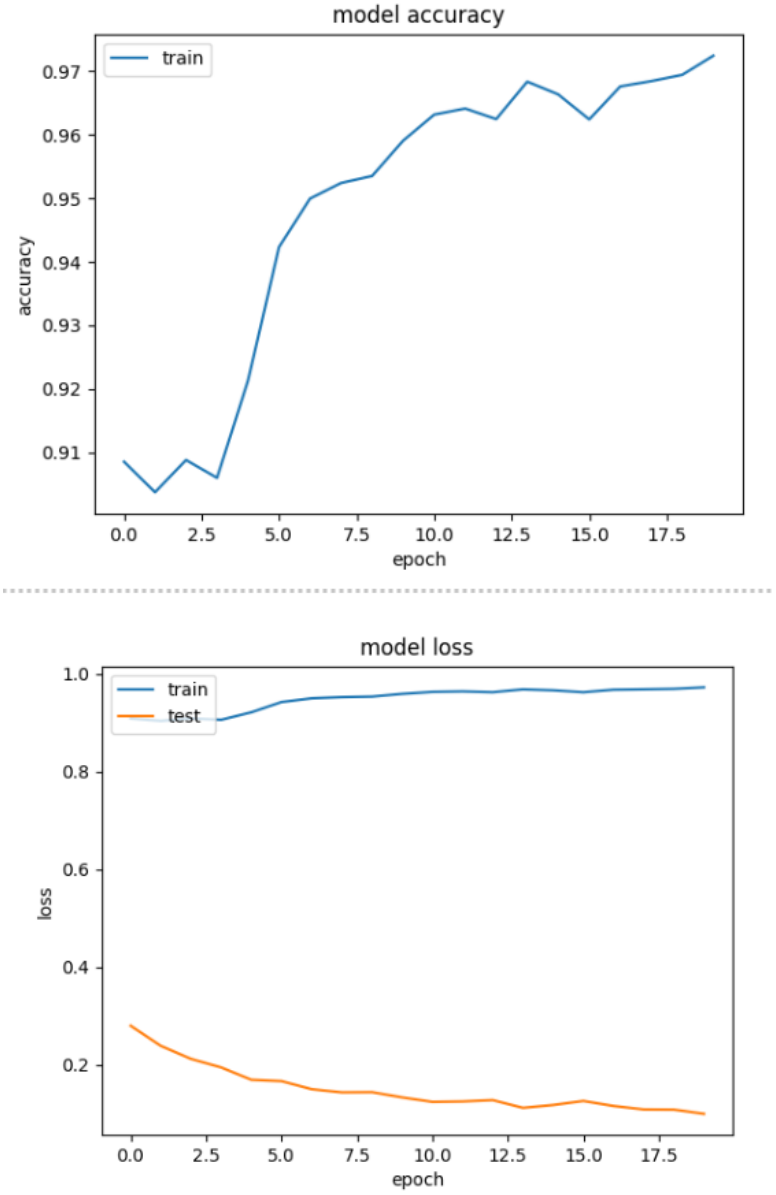


Figure 3.9 Graph of Unet training accuracy (top) and loss (bottom)

with many lines (middle row).

3.5.2 Rail Detection

As mentioned earlier, rail detection was accomplished through the combination of edge detection and line detection. A comprehensive comparison was conducted,

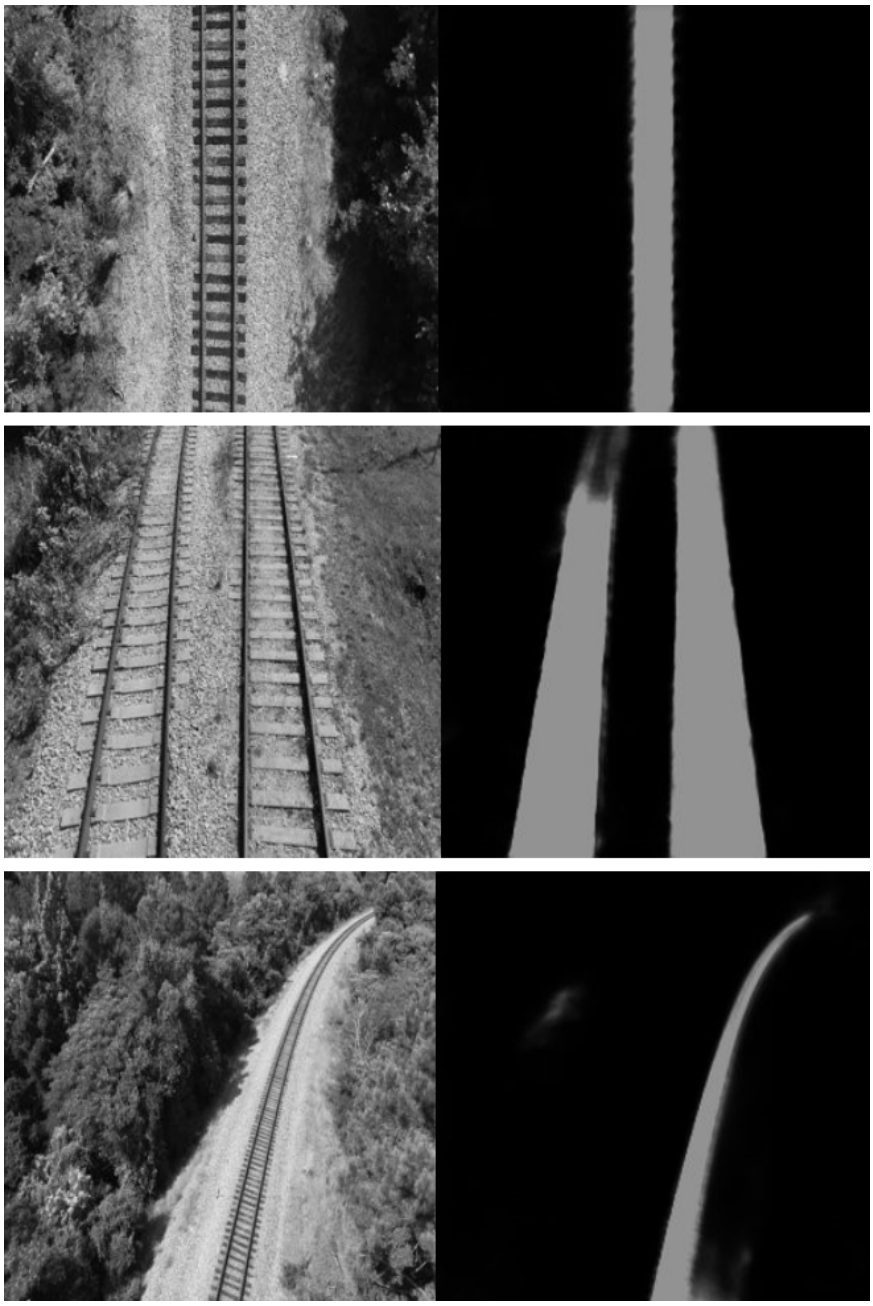


Figure 3.10 Examples of track region detection using Unet

exploring all possible combinations of three widely used edge detection algorithms (Canny, Sobel, and Laplace) and three popular line detection algorithms (FLD, LSD, and Probabilistic Hough Transform). The evaluation criteria comprised two key metrics: false positives and detection percentage. False positives quantified the frequency

with which the system identified lines as rails that did not belong to the actual rail; this metric was computed by determining how many detected lines fell outside the ground truth rail area, with lower values considered more favorable. Detection percentage provided an estimate of the system's accuracy in detecting the ground truth rail area, calculated by determining the adjusted percentage of detected lines within this area. The results reveal that certain combinations of edge detection and line detection methods excelled in detection percentage but performed poorly in false positives and vice versa. The full result charts can be seen in Fig. 3.11. Remember that the goal is for a combination to have a high detect percent and a low false positive. Based on this, the Canny - FLD combination ranked among the top two in both metrics. For this reason, it was chosen to be used in the final system. The results were relatively close, however, and a clear decision on the best combination was not present. This supports the mixed evaluations seen in the literature as explored in the background section above.

3.5.3 Rail Interpretation

As elaborated in the previous section, some aspects of the interpretation phase can only be thoroughly assessed during the practical implementation of the system on a drone. With this in mind, both the vanishing point approach and the trajectory approach were employed in this early, offline stage of implementation, serving primarily as a proof of concept and setting the groundwork for potential improvements. While the recommended command proved accurate for most frames in both approaches, a few issues surfaced. Specifically, challenges were encountered in the trajectory calculation method, where obstructed or misidentified sections of the rails led to skewed trajectories. A similar issue occasionally arose in the vanishing point approach, where one of the rail lines was incorrectly identified, significantly distorting the vanishing point. However, these issues occurred sporadically and could be mitigated by filtering

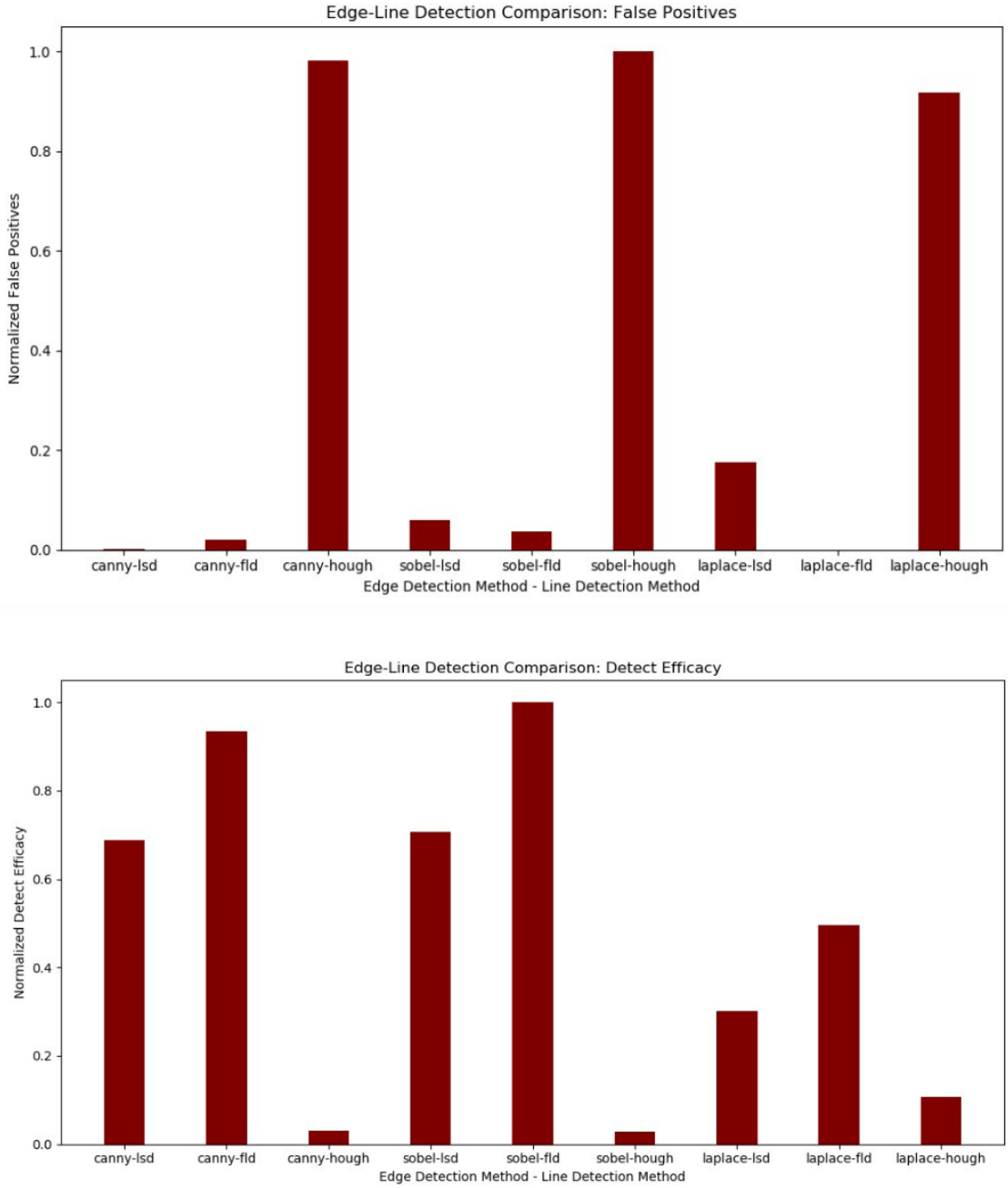


Figure 3.11 Edge and line detection comparison graphs

outliers when comparing adjacent frames. In terms of future work, practical implementation of the TrackNet system on a drone platform is crucial for evaluating its real-world performance. A thorough comparison of track interpretation implementations, development of an adaptive system, and integration with control systems

are identified as areas for further exploration. The developed TrackNet system was designed with this in mind, and it is ready to be implemented onboard a drone system.

3.6 Conclusion

This paper introduces TrackNet, a novel system for railroad track detection and tracking. It systematically reviews existing methodologies, categorizing them into Region Detection, Rail Detection, and Compound Methods. Region Detection methods, including HSV color extraction and SVM classification, prioritize preserving the entire track area. Rail Detection explores edge and line detection methods, informing the proposed TrackNet system. TrackNet integrates both region and rail detection, utilizing Unet for region detection and edge/line detection for rail detection. The paper then extends the TrackNet system, detailing interpretation methods including vanishing point and trajectory detection to prepare for a fully integrated, track-following drone. Future work involves implementing TrackNet on a drone, interpretation approach comparisons, adaptive system development, and integration with onboard flight control systems.

Chapter 4

Track Following and UAV Implementation

Although TrackNet, as outlined in Chapter 3, is a novel approach to track detection compiling the best methodologies seen in the literature (Chapter 2), the primary contribution of this work is an online implementation of this system for experimental testing and comparison of these methods in live track following. The following sections will explore the process of employing the TrackNet system onboard a customized DJI Matrice 100 UAV and utilizing it for high level flight controls.

4.1 Track Interpretation

Between the completion of track detection and the initiation of track following, a few tasks must be completed. The system needs to interpret the rail lines and determine the flight maneuvers required for the UAV to follow the track. As described earlier, these interpretation methods are contingent upon the configuration of the camera onboard the UAV system. This divergence prompted the development of two distinct approaches for the sake of comparison, namely a forward-facing camera approach and a downward-facing camera approach. As explored in Chapter 2 and 3, these are the two approaches seen in the literature and utilize vanishing point detection and trajectory detection, respectively.

In the case of a forward-facing camera, the vanishing point is assumed to be a reasonable approximation of the track's direction. As explained previously, this point is determined by identifying the intersection point of the rail lines. However, implementing this in TrackNet presents a challenge due to how rails lines are detected.

There are several methods for this process in the literature and based on these the decision was made to base vanishing point calculations on the longest straight line segments in each rail element. This was because of an assumption that the lines closest to the UAV will provide a better indication of the current direction of the track. Perspective causes further objects to be viewed as smaller, therefore the most pertinent lines of the rail should be those that are perceived as longest. To achieve this both rail elements (either single lines or chains of lines) are sorted by length. The longest lines within each element are characterized by two points, $p_{i1} = (x_{i1}, y_{i1}), p_{i2} = (x_{i2}, y_{i2})$, with a slope, $m = \frac{y_{i2}-y_{i1}}{x_{i2}-x_{i1}}$ for any line i. Using the point-slope form of a line, two equations can be created for the longest lines and the vanishing point can be determined by finding the solution of a system of the following two equations (4.1, 4.2):

$$y - y_{i1} = m_i(x - x_{i1}) \tag{4.1}$$

$$y - y_{j1} = m_j(x - x_{j1}) \tag{4.2}$$

Where (x, y) is the vanishing point. This method has the potential to be susceptible to false positives if the longest line in either element is itself a false positive. A more complex version of this selection process could be developed, however, in testing the TrackNet system has proven to be sufficient at filtering these false positives out before vanishing point calculation. In following tests, this filtering is shown to be adequately effective for the designed control system. An example of vanishing point detection can be seen in Fig. 4.1 where the vanishing point is visualized as a green dot at the center of the top of the image. Once the vanishing point has been calculated, the forward-facing system is prepared to initiate the track following process.

On the other hand, utilizing a downward-facing camera necessitates a different approach. Here, the objective is to determine the centerline of the track based on the rail lines. For this, each rail element was divided into a series of points, including

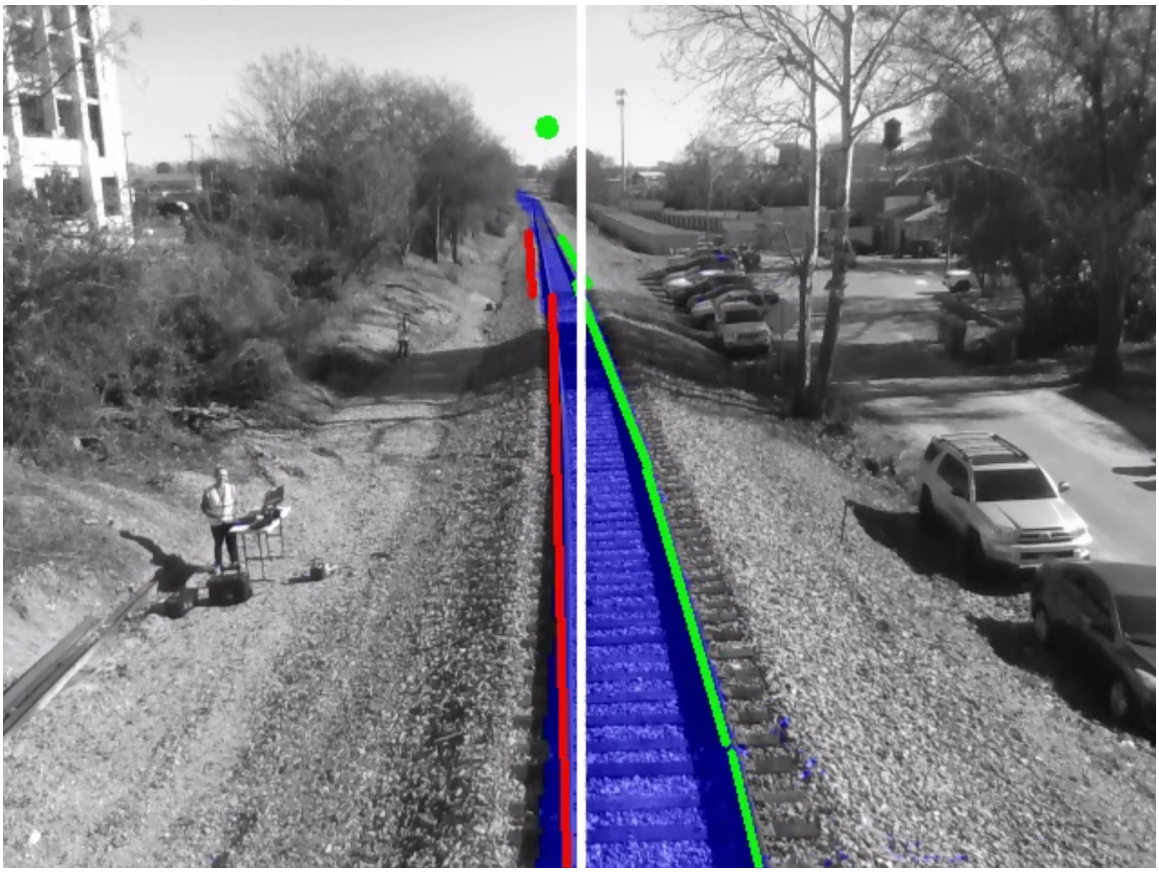


Figure 4.1 Example of vanishing point detection. The white line is the center column of the frame. The red and green lines are the rail lines. The green dot is the vanishing point.

the endpoints and equally spaced points along each line, as depicted by white dots in Fig. 4.2. Subsequently, for each rail, a new curve was generated through quadratic interpolation of these points. This process decreases the influence of false positives as it equally weights all rail lines according to their length. This means that if a false positive line is a part of the rail, it's influence will be overshadowed by the other, correct lines. This solution also allows midline calculation in gaps between lines along the same rail. Once these new rail line approximations are calculated, the midline between them is determined creating a trajectory line at the center of the two rails. For this, a midpoint is calculated for every row of pixels, y_i, using:

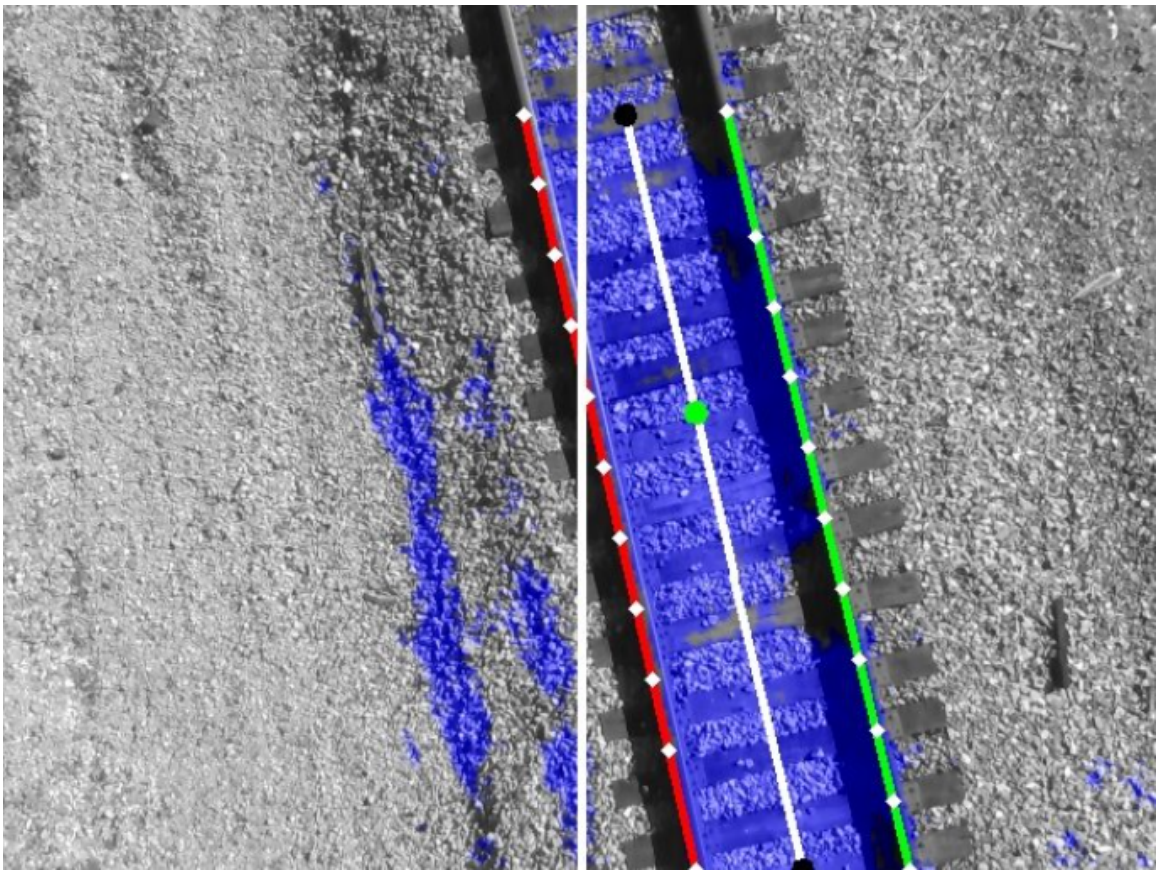


Figure 4.2 Example of trajectory detection for downward-facing camera approach. The green and red lines are the detected rail lines. The vertical white line is the center column of the frame. The angled white line is the calculated trajectory. The black points are the end points of the trajectory. The green point is the closest point of the trajectory.

$$y_i = f_1(x_i) + \frac{(f_2(x_i) - f_1(x_i))}{2}, f_1(x_i) < f_2(x_i) \tag{4.3}$$

Where $f_1(x_i)$ and $f_2(x_i)$ are the functions of the quadratic interpolations of each rail line. These points are then plotted creating a trajectory curve. This curve is represented as a white line in Fig. 4.2. In practice, the current implementation of this method occasionally produces skewed trajectories when fewer lines are identified for either rail. In spite of this, as long as at least one rail is identified, the midline will always be somewhere on the track if not in the perfect center. This means that

in the worst case, when only one rail is identified, the midline will simply be that rail and the UAV will still be capable of flying along the track. This process successfully bridges the gap between track detection and track following.

4.2 Track Following: Forward-Facing Approach

Once the detected tracks have been successfully interpreted using either the forward-facing or downward-facing approach, the final step is to translate this data into commands to be sent to the UAV. Again, this process will be similar but will diverge depending on the camera orientation. In either case, however, the elements discussed in Chapter 2 will need to be determined: the current state of the UAV (or process variable $PV(t)$), the desired state of the UAV (or setpoint SP), the difference (or error) between these two states, and the change required to decrease this error. The difference in approach will lie in what these elements are and how they are calculated.

For the forward-facing camera approach, these elements will revolve around the vanishing point. The process variable is determined to be the the center column, x_c, of the camera's frame, as it is assumed to indicate the direction of the UAV's heading (visualized in Fig. 4.1 as a white line). For the camera in this implementation the frame size is 640 px by 480 pixels, therefore $x_c = 320$. The set point of the UAV will be the horizontal column of the vanishing point, $x_{vp}(t)$, as it provides an estimation of the track's direction. It is presumed that if these two values, x_c and $x_{vp}(t)$, are equal, the direction of the UAV's travel will be the same as the direction of the track. The direction error, $e_d(t)$, between the desired and current states is thus calculated as the horizontal distance between the center of the camera's frame and the vanishing point:

$$e_d(t) = x_c - x_{vp}(t) \tag{4.4}$$

If this error is zero, the UAV should be traversing in the same direction as the track's path, when given a constant forward velocity. Consequently, the change required to decrease this error involves adjusting the UAV's yaw, by generating angular acceleration about the vertical axis of the UAV. If the vanishing point is on the right half of the camera frame, the UAV needs to yaw to the right, and vice versa. The magnitude of this yaw is determined by the magnitude of the error between the vanishing point and the center of the frame. In the developed system, this error, $e_d(t)$, is normalized and input into a PD controller as follows:

$$e_{d,norm}(t) = \frac{e_d(t)}{x_f} * c_{max,yaw}, \tag{4.5}$$

Where $e_{d,norm}(t)$ is the normalized error value, x_f is the pixel width of the frame, and $c_{max,yaw}$ is the maximum feasible value of yaw control effort. This normalized error is then used in the PID control formula, equation 4.6. This formula is used for all of the controllers that are to follow with differing values for the gains (K_p, K_d, K_i). In the cases where a PD controller is used, the integral gain (K_i) is simply set to zero and similarly PI controllers set the derivative gain (K_d) to zero. This fundamental formula is as follows:

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} + K_i \int_0^t e_{p,norm}(t), \mathrm{d}t. \tag{4.6}$$

Where $u(t)$ is the command effort output, $e(t)$ is the error, and K_p, K_d and K_i are the proportional, derivative and integral gains, respectively. In this case, $e(t) = e_{d,norm}$ and $u(t)$ is the yaw control effort. The gains of this controller were then determined experimentally resulting in a $K_p = 0.75$, $K_d = 0.458$ and $K_i = 0$, producing a PD controller. The tuning process began by utilizing the zeigler-nicholes method for tuning PID controllers, then was adjusted after additional testing, this process is further detailed in Chapter 5. These preliminary tests determined that

the integral portion of a PID controller was unnecessary for this application, but the derivative gain was significantly more important to resolve a large oscillation in the early implementation. The output of the controller is then directly sent to the UAV's yaw velocity through the DJI software development kit (SDK) robot operating system (ROS) package as explained in Chapter 5.

This simple yaw control system proved effective originally, but a flaw was exposed in further testing, particularly when the UAV encountered long and curving tracks. This issues is similar to that mentioned in the evaluation of [3] in Chapter 2, namely that direction control is not always sufficient. This first type of control ensures that the UAV is traveling in the same direction as the track is from a visual perspective, however, it does not directly control the horizontal position tangent to the track's direction. In the experimentation in Chapter 5 it was seen that as the UAV traverses along the track it could accumulate a positional error and no longer be directly above the track region. Although the camera would percieve this because the rail lines would drift to one side of the frame, the vanishing point would not change position accordingly. For this reason and additional layer of control was developed to solve this problem. The objective of this system is to perceive this drift of the rails and to correct it utilizing the roll channel of the UAV. For this the endpoints of all detected lines included in the rail elements (single lines or line chains) were sorted by pixel row, y, and the point with the largest value (with origin located at the top left corner of the frame) for each rail was selected, $a_i = (x_i, y_i), a_j = (x_j, y_j)$. The midpoint, $p_m = (x_m, y_m)$, between these two points was then calculated:

$$x_m = x_i + \frac{x_j - x_i}{2}, x_i < x_j \tag{4.7}$$

$$y_m = y_i + \frac{y_j - y_i}{2}, y_i < y_j \tag{4.8}$$

This point is then assumed to be the center of the track because it is the midpoint

between the two rail lines. Additionally, the lowest point of the rails is used as it is assumed to represent the section of track closest to the UAV. Now, if this point drifts away from the center of the frame, it represents the horizontal position error explained above. Here the process variable is again the center column of the image, x_c, the set point is the horizontal value of this midpoint, x_m, and the error can be modeled and normalized similarly as follows:

$$e_p(t) = x_c - x_m(t) \tag{4.9}$$

$$e_{p,norm}(t) = \frac{e_p(t)}{x_f} * c_{max,roll}, \tag{4.10}$$

Where $e_p(t)$ is the position error, $e_{p,norm}$ is the normalized position error, and $c_{max,roll}$ is the maximum control effort feasible for the roll channel. This output is then channeled through a PI controller modeled using equation 4.6 where $e(t) = e_{p,norm}$ and $u(t)$ is the roll control effort.

This controller was tuned in the same way as the yaw controller, but during this testing the derivative gain proved unnecessary and a persistent steady state error indicated importance of the integral gain. The gains, K_p and K_i were therefore tuned to 0.25 and 0.1, respectively. With this, a completed track following system with a forward-facing camera was developed and ready for implementation and testing onboard a UAV.

4.3 Track Following: Downward-Facing Approach

In many ways the controls system of the downward-facing approach is similar to that of its forward-facing counterpart with differing inputs. In this scenario, the current state of the UAV is still approximated as the horizontal center of the camera's frame because it remains the direction in which the UAV is traversing and the desired state is the trajectory line. As is explained in Chapter 2, two dimesions of control can be

applied to these states: the direction of travel and the horizontal position over the track. For the direction control, the vertical line along the center of the frame is the process variable correlating to the direction the UAV is traveling projected onto the track's plane. The trajectory line is then utilized as the directional set point, however, this causes an issue due to the fact that the trajectory is not necessarily a straight line. To adjust for this, the trajectory direction is calculated based on the slope between the top point of the trajectory and the bottom point. Namely, every point in the trajectory is sorted by its y value and the maximum and minimum values are selected. The line between these two points is then utilized as the set point. With this, the direction error, $e_d(t)$, can be calculated as the angle difference between this trajectory line and the center line of the frame as follows:

$$e_d(t) = arctan(\frac{[(x_{t,1} - x_{t,2}) * (y_{c,1} - y_{c,2})] - [(y_{t,1} - y_{t,2}) * (x_{c,1} - x_{c,2})]}{[(x_{t,1} - x_{t,2}) * (x_{c,1} - x_{c,2})] + [(y_{t,1} - y_{t,2}) * (y_{c,1} - y_{c,2})]}), \quad (4.11)$$

Where the two endpoints of the trajectory line are $t_1 = (x_{t,1}, y_{t,1}), t_2 = (x_{t,2}, y_{t,2})$ and the top and bottom point of the frame's center column are $c_1 = (x_{c,1}, y_{c,1}), c_2 = (x_{c,2}, y_{c,2})$. This direction error serves as one degree of freedom and controlling it to zero theoretically ensures that the UAV is traveling in the same direction as the track. This error is then normalized as in equation 4.10. This normalized error is input into a PD controller modeled according to equation 4.6 that was experimentally tuned as above with final gains of $K_p = 1.05$ and $K_d = 0.2$. This controlled output is utilized in the flight controller's yaw channel.

Even with this yaw controller, the UAV's horizontal position may still accumulate error. This error can be approximated as the horizontal distance between the center point of the camera frame (the process variable) and the closest point on the trajectory line (the set point). If this distance is controlled to zero, the UAV will be successfully positioned above the track. To calculate this error, the distance between the center

point of the camera and every point of the trajectory is calculated using the distance formula and the point with the minimum of these distances is taken as the error:

$$e_p(t) = argmin(\sqrt{(x_c - x_{t,i}(t))^2 + (y_c - y_{t,i}(t))^2}), \tag{4.12}$$

Where $e_p(t)$ is the positional error at any time, the center point of the frame is $p_c = (x_c, y_c)$ and any point, i, along the trajectory is $p_t = (x_{t,i}(t), y_{t,i}(t))$. Again, this error is utilized in the PID controller equation 4.6, experimentally tuned and input into the roll channel of the flight controller. The final gains for this controller are $K_p = 0.75$ and $K_i = 0.2$.

With both dimensions of error being controlled to zero it is assumed the UAV will successfully follow the track if given a constant forward velocity. This successfully outlines a fully formed system that receives a video input, detects the track, and determines controls to follow the track. At this point the final step is to implement this system onboard a UAV.

4.4 Drone Implementation

The above system functions in theory, but must be implemented onboard a UAV to perform proper testing and to evaluate its effectiveness. The UAV chosen for such implementation was a DJI Matrice 100, a quadrotor equipped with an integrated flight controller, DJI N1. This flight controller comes with a software development kit (SDK) for developmental purposes. Additionally, the Matrice 100 comes equipped with the DJI guidance system, which enhances stabilization, especially in GPS-denied environments. In our implementation, the guidance system was utilized with only one, downward-facing camera unit. The N1 flight controller utilizes the sensors from this camera for low-level flight controls, however, this data is not able to be utilized for our high-level controls. For this reason, the guidance system has no effect on the track

detection and following process, but only in the UAV's internal flight controls. The system does provide another sensing system to support steady flight regardless of GPS availability. Mounted on the UAV's frame is an Intel NUC 11 Performance Mini PC Kit, providing onboard processing power, along with an Intel RealSense D435 Camera. These components are affixed to the UAV's frame using custom 3D-printed parts. The widely used Robot Operating System (ROS) framework facilitates communication among these three primary devices, specifically utilizing the RealSense2camera ROS package and the djisdk ROS package, along with a custom TrackNet ROS package developed for this application. This package performs the function of processing the RealSense camera's video and sending control outputs to the Matrice's flight controller. The process utilizes two ROS nodes, one for TrackNet processing and one for sending the control signals. The TrackNet processing node subscribes to the RealSense2camera's /camera/image_raw topic, acquiring the video feed. These images are then processed by the TrackNet system, and control outputs are calculated. This process is completed at around 3.5 frames per second and the control effort is published to a ROS topic. The control node monitors this topic and sends the control effort to the Matrice's flight controller. The method of control that is used is based on velocity control provided by the djisdk package in the form of ROS services. The control node sends a continous stream of forward (x) velocity commands to the associated ROS service to maintain a constant forward velocity. When the TrackNet node publishes yaw or roll control effort, this is added to the ROS service call. With these two nodes, the Matrice is given a constant forward velocity and its direction and position are adjusted as TrackNet is able to process the RealSense's video stream. An image of the complete Matrice 100 setup is provided in Fig. 4.3.

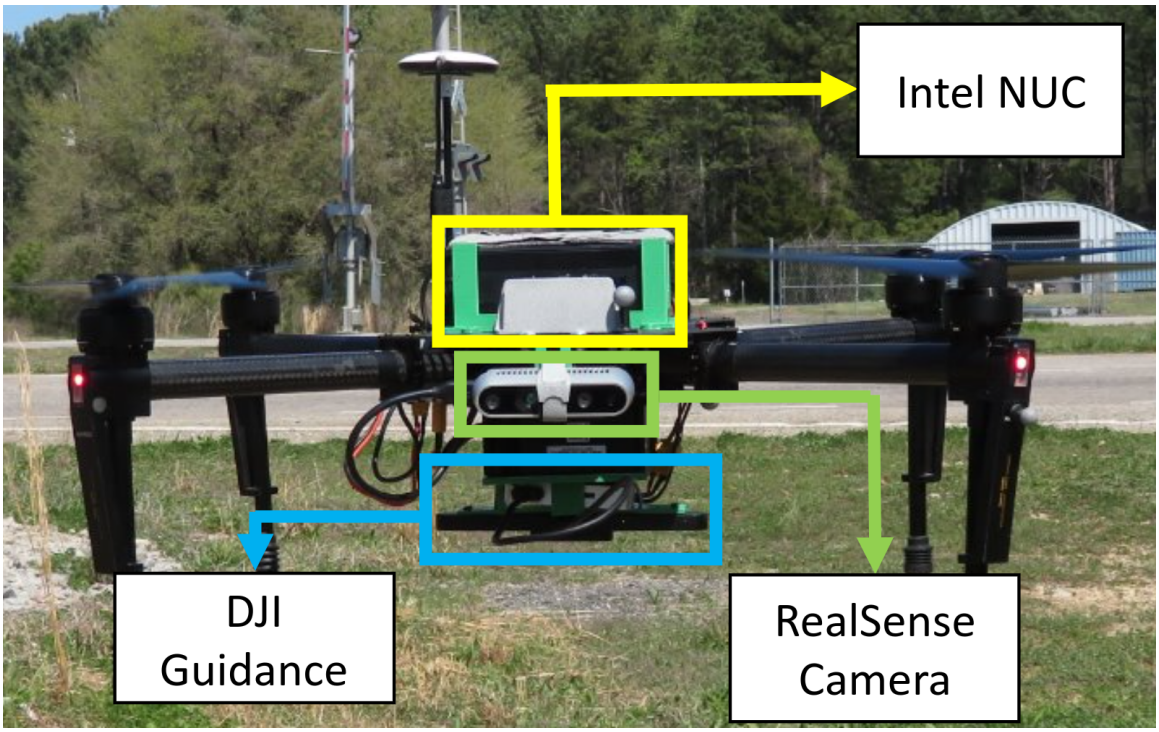


Figure 4.3 DJI Matrice 100 equipped with Intel NUC 11, Intel RealSense D435 camera, and DJI Guidance system.

Chapter 5

Results

With the complete TrackNet system developed and tested offline on prerecorded videos, as explained in Chapter 3, testing of the implementation onboard the UAV system was the next stage of experimentation. This includes both the forward-facing camera and the downward-facing camera approaches. For each approach, several stages of testing were conducted, starting with stationary indoor tests to establish proof of concept and to provide initial gains for each controller. This series of tests was conducted inside the Unmanned System and Robotics laboratory at the University of South Carolina. Simple images of tracks, from both a forward-facing perspective and a downward-facing perspective were utilized as a simulated track environment. Subsequently, these systems were tested outdoors on a real track to validate their efficacy as well as in effort to determine issues in the first versions of the systems. These experiments took place along a 45 meter section of railroad situated in Columbia, South Carolina, next to the University of South Carolina's athletic center. An image of this section of track, taken from Google Maps can be seen in Fig. 5.1. The track was simple and consisted of mostly straight track, located in an urban environment with nearby buildings and power lines. After learning from these initial tests, more extensive outdoor experiments were carried out at the South Carolina Railroad Museum in Winnsboro, South Carolina. This track contained varying track configurations, including a sharp curve in both directions as well as a longer straight section and was situated in a more rural, forest environment and a image of this location can be seen in Fig. 5.2. In over several experiments two sections of this track were utilized at dif-

ferent times and these sections are shown in Fig. 5.2. The shorter section began with a straight track followed by a left curve and was a total of 165 meters. The longer section began with a right curve then a straight section followed by a left curve and was a total of 300 meters.

Figure 5.1 This is an image of the railroad section used for experiments near the University of South Carolina's Athetic Village.

The following sections of this chapter will delineate the results obtained from this series of testing and will outline the findings related to both approaches explained in Chapter 4. In all such tests, two types of error were determined and will be seen. Firstly, there are the several instances of direction error and position error. These values are calculated purely based on track detection and will differ depending on the type of track interpretation. These errors are defined in Chapter 4, namely $e_{d,norm}(t)$ and $e_{p,norm}(t)$, and are utilized as the input to their corresponding controllers. In

Figure 5.2 This is is an image of the railroad section used for experiments at the South Carolina Railroad Museum in Winnsboro, South Carolina. Two sections of this track were used and are labeled in the figure as 'short section' and 'long section'.

attempt to make the scale of these values more easily comparable, they have been converted to a percentage of the max feasible control effort. These values were experimentally determined to be $c_{max,yaw} = 50deg/s$ and $c_{max,roll} = 1m/s$. This measure of error is helpful to show the efficacy of the control system itself, but is not valid in providing a meaningful quantification of the system's success in following the track. For this purpose, a separate track distance error was calculated in reference to the track's location itself, $e_t(t)$. This value was calculated as the distance between the UAV's horizontal position and a ground truth position of the center of the track:

$$e_t(t) = argmin(\sqrt{(x_{UAV}(t) - x_{t,i}^2 + (y_{UAV}(t) - y_{t,i}^2)}, \tag{5.1}$$

Where $(x_{UAV}(t), y_{UAV}(t))$ is the position of the UAV at any given time as determined by the odometry of the UAV and $(x_{t,i}, y_{t,i})$ is every position, i, along the track as determined by GPS coordinates found on Google Maps. This provides a measure of the accuracy of the UAV's flight when compared to the track and therefore a way

to determine the system's effectiveness. This position is measured in meters and as a reference, the distance between track rails (track gauge) is approximately 1.5 meters.

5.1 Forward-Facing Experimentation

In employing the forward-facing camera approach, the initial system utilized only a PD controller on the yaw velocity channel, as outlined in Chapter 4. The roll controller was not initially implemented because there is no example of such a system in the literature. Later the need for this system became prevalent and it was integrated as will be explained. Because of this, the initial implementation was an adaptation of the most common method found in the literature as can be seen in Chapter 2, seeking to detect the track, calculate the vanishing point, and control the yaw velocity.

5.1.1 Experiment 1

Initially, nine trials were conducted using this system at the University of South Carolina location. For these trials, the PD controller was tuned to the following gains: $K_p = 0.9$ and $K_d = 0.25$. Of the nine trials, the UAV successfully followed the track and it never lost sight of the track. The system demonstrated its capability to effectively track a straight railroad using only camera input and onboard processing. It illustrated that the TrackNet system functions in real-time, even on tracks it was not trained on. Furthermore, it showcased that the processing requirements could be met entirely online and onboard the UAV, achieving an average rate of 3.5 frames per second, sufficient for maintaining the UAV's position above the track at an initial forward velocity of one meter per second. Across these nine intial trials, the average directional error was 15.409%. The results for each trial are laid out in Table 5.1 where the average directional error for each trial is shown. Scrutiny of these results revealed a significant, continuous oscillation in the yaw control system. This can be seen in Fig. 5.3, where the vanishing point error is plotted alongside the command

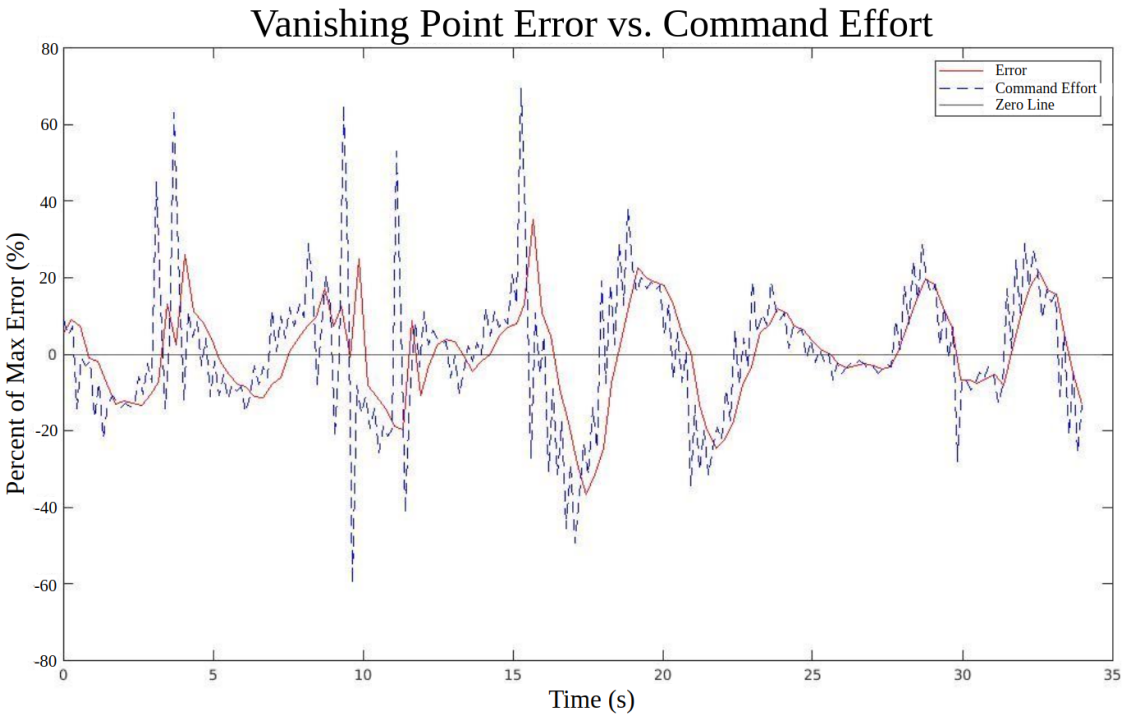


Figure 5.3 Example of track following test from the first experiment with forward-facing camera configuration. Graph shows normalized vanishing point error (solid red line) and related yaw command effort (dotted blue line) across time.

effort. The problematic oscillations in the system are clear, with peak amplitudes around 36%.

As no systematic cause was able to be detected, it seemed that the PD gains simply needed to be retuned to be effective over an actual track. Because the issue was an oscillation, the primary solution was a redution in the proportional gain. Additionally, the derivative gain was increased to compensate resulting in final values being $K_p = 0.75$ and $K_d = 0.458$.

5.1.2 Experiment 2

With these adjustments made, a second round of tests was conducted at the same location. This time, a series of eleven trials were executed to evaluate the improvements made in the controls system as well as to test the systems ability at differing

Table 5.1 Forward-Facing Experiment 1: TrackNet Errors

Trials:	**Direction Error (Vanishing Point):**
Trial 1	8.1587%
Trial 2	26.2635%
Trial 3	7.4105%
Trial 4	10.0892%
Trial 5	18.7034%
Trial 6	11.1638%
Trial 7	20.9608%
Trial 8	26.1773%
Trial 9	9.7574%
Average:	**15.409%**

altitudes. The results of each trial are outlined in Table 5.2 and it can be seen that the average vanishing point error overall improved to be 7.8054%. It can also be noted, that the values of error for each trial seem to vary widely. This is however explained by the fact that these trials were conducted in alternating directions along the track. This explains why the values seem to alternate in magnitude. It seems that the system performed worse when it was traveling in towards a curve that was beyond the flight path, but within the camera's view. What is more interesting is the lack of change in detection accuracy with altitude. Across the three tested altitudes (two meters, four meters, and five meters) there is not a significant difference in error values except in the five meter case. This variance, however, is not hugely significant and is only across two trials. Because of this, it seems that the forward facing approach's detection ability is not significantly affected by the current altitude. This is to be expected because the system should work in the same way as long as the track is visible. The only major difference that altitude creates is the minimum track distance that is visible within the frame. In spite of this, it is clear from these trials that issue of oscillations in the yaw controls was greatly improved even in varying situations.

An example of this can be seen in Fig. 5.4 where the peaks of the minor oscil-

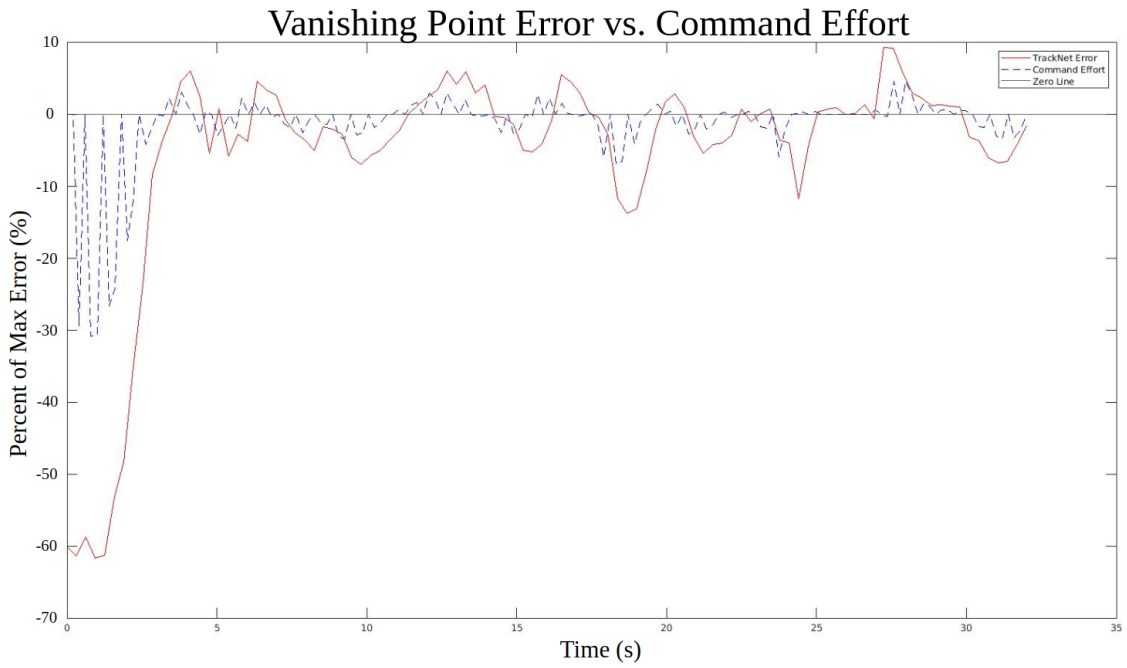


Figure 5.4 Example of track following test from the second experiment with forward-facing camera configuration. Graph shows normalized vanishing point error (solid red line) and related yaw command effort (dotted blue line) across time.

lations are now approximately 15%. This shows a great improvement in the control system's ability to track the vanishing point and maintain the correct direction of flight, however, during these experiments a potential issue was subjectively recognized. In spite of the improved controls, the UAV's position above the track seemed to be worse than the previous experiment. A horizontal drift in the UAV's position was noticed, but was not seen in the directional error statistics. The following experiments further explore this potential issue and eventually determine that this was a product of the flaw in simple yaw controls for the forward-forward facing approach as explored in Chapter 4. This is further explained in the next experiment. After this series of results, the roll control system outlined in Chapter 4 was developed and implemented, seeking to solve this problem. The PI controller gains for the roll controls were experimentally determined to be as follows: $K_p = 0.25$ and $K_i = 0.1$.

Table 5.2 Forward-Facing Experiment 2: TrackNet Errors

Altitude:	**Trials:**	**Direction Error:**
2 m	Trial 1	4.2962%
2 m	Trial 2	12.1520%
2 m	Trial 3	5.4814%
2 m	Trial 4	11.0920%
2 m	Trial 5	2.6484%
2 m	**Average:**	**7.1340%**
4 m	Trial 1	10.2919%
4 m	Trial 2	4.4333%
4 m	Trial 3	10.2798%
4 m	Trial 4	3.5532%
4 m	**Average:**	**7.1395%**
5 m	Trial 1	11.0920%
5 m	Trial 2	10.5397%
5 m	**Average:**	**10.8159%**
Overall	**Average:**	**7.8054%**

5.1.3 Experiment 3

The next phase of testing transferred to the second location at the South Carolina Railroad Museum (Fig. 5.2). Here, the two implementations of the forward-facing approach were tested and compared, one with only yaw controls and the other with yaw and roll controls. The difference in these approaches is significant and obvious as the yaw-only implementation failed to traverse the curve in the track. This system was tested along a section of straight track followed by a sharp left turn (Box labeled "short section" in Fig. 5.2). It is clear that as the UAV was traversing the straight section, it begins to drift to the right side of the track. This could be due to a strong wind from the left that was present during these tests or simply due to noise in the system that is not corrected for. Either way, the thing to note is that the directional error is not significantly effected by this horizontal drift and so the UAV does not correct. This is clear in Fig. 5.5.

Here, the plot of the TrackNet error and command effort is set alongside the flight

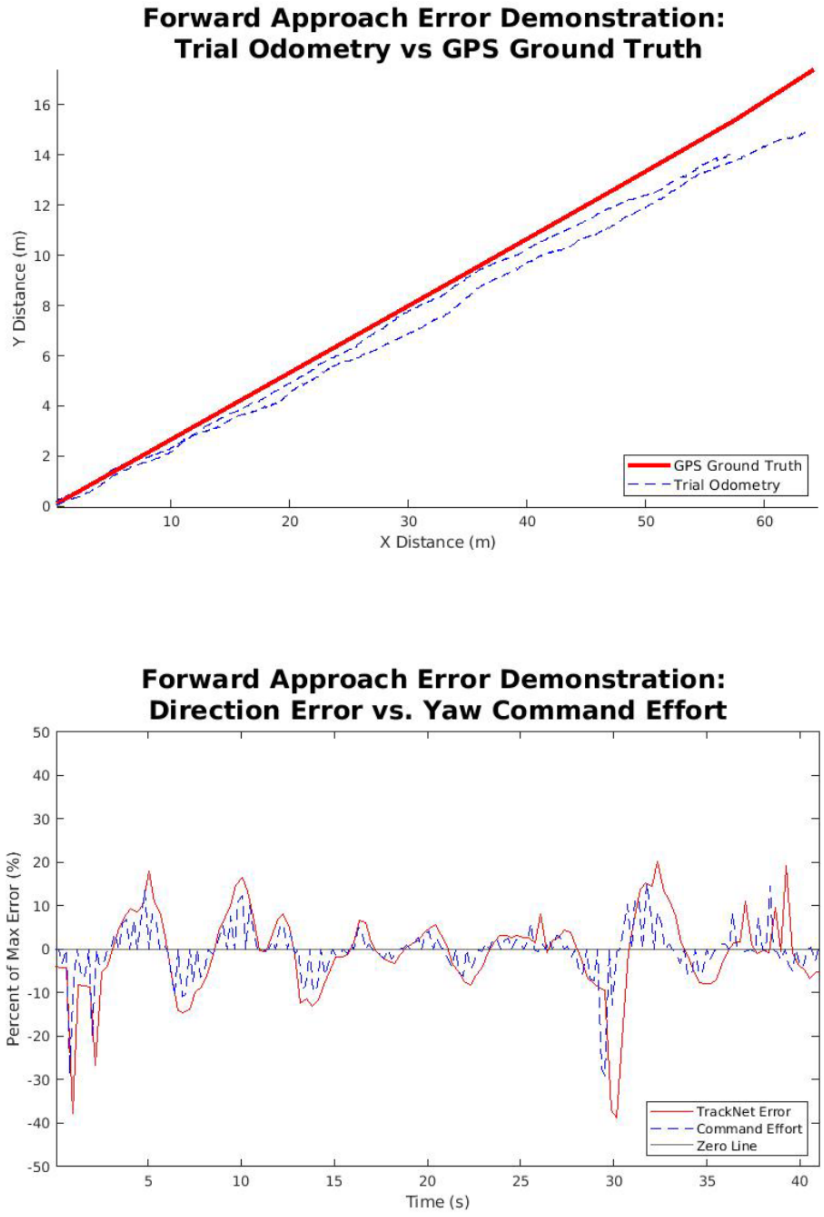


Figure 5.5 Visual explanation of the issue in vanishing point control using only the yaw channel. Comparison of vanishing point error across time (bottom) and track position error along length of track (Top).

path plot. One can see that as the UAV traverses along the track (left to right in both plots), the UAV drifts away from the track's position, but the vanishing point error does not significantly change. This proves the vanishing point's inability to recognize

and adjust for a positional drift as explained in Chapter 4. This drift becomes an additional issue as the UAV reaches the curve and fails to recognize the need to turn fast enough. To avoid crashing, the system had to be aborted in each trial. This error is one that has not yet been identified in the literature as the majority of systems outlined simply perform this directional control, but fail to consider this problematic drift. This is the reason that the roll controller was developed as outlined in Chapter 4 and tested as a successful solution to this problem.

The forward-facing approach utilizing both yaw and roll controls (as outlined in Chapter 4) was also tested to identify if this drifting issues was still present. This system was tested on a larger section of track at the Railroad Museum location (box labeled "long section" in Fig. 5.2), containing a curve to the right, a straight section, and a curve to the left. This improved system successfully traversed this entire section of track maintaining an average track position error of 1.9766 meters and TrackNet errors of 3.7694% for directional error and 6.3267% for positional error. It can be seen in the flight paths in Fig. 5.6 that although a slight positional drift occurs, it is corrected by the roll control system.

Table 5.3 Forward-Facing Experiment 3: Track Distance and TrackNet Errors

Forward Vel.:	**Trials:**	**Track Dist.:**	**Dir. Error:**	**Pos. Error:**
1 m/s	Trial 1	2.0288 m	4.2349%	11.0287%
1 m/s	Trial 2	2.0843 m	3.5404%	6.7021%
1 m/s	Trial 3	2.1226 m	4.8564%	6.2240%
1 m/s	**Average:**	**2.0786 m**	**4.2106%**	**7.9849%**
2 m/s	Trial 1	2.0171 m	3.5751%	6.3422%
2 m/s	Trial 2	2.1210 m	3.4121%	5.6712%
2 m/s	Trial 3	1.9459 m	3.3795%	6.0406%
2 m/s	**Average:**	**2.0280 m**	**3.4556%**	**6.0180%**
4 m/s	Trial 1	1.7772 m	3.4622%	5.9298%
4 m/s	Trial 2	1.6113 m	4.0003%	4.5682%
4 m/s	Trial 3	2.0813 m	3.4636%	4.4337%
4 m/s	**Average:**	**1.8233 m**	**3.6420%**	**4.9772%**
Overall	**Average:**	**1.9766 m**	**3.7694%**	**6.3267%**

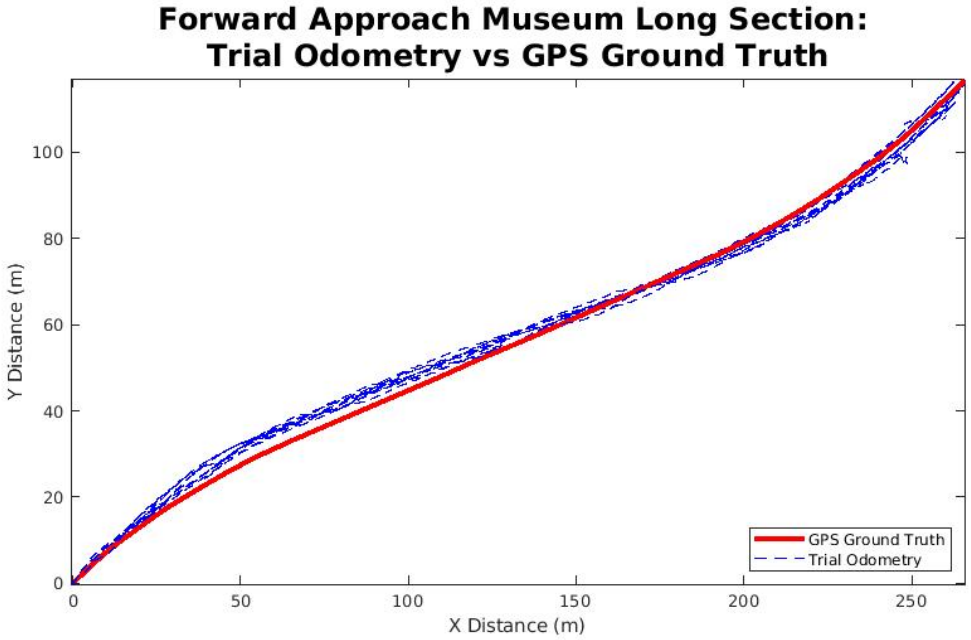


Figure 5.6 Improved forward-facing approach utilizing roll controls in addition to yaw controls. Flight path seen in dotted blue line and GPS ground truth seen in solid red line.

This experiment also tested the system's performance at varying forward velocities, namely one, two and four meters per second. The outline of all trial results can be seen in Table 5.3. Also, whisker plots of each trial can be seen in Fig. 5.7, where the center line of each box is the median, the top of the box is the 75th percentile, the bottom of the box is the 25th percentile, the top and bottom lines reach to the maximum and minimum values not considered outliers, any red dots are the outliers, and the horizontal line is the total average over all trials. From this it is clear that the system is not greatly affected by increasing forward velocity and performance actually seems to increase with speed. This trend seems counter intuitive, however, there is a potential explanation. To understand the situation it is important to recognize that as forward velocity increases, the distance between control updates also increases. This is because the processing power onboard the UAV is still at a constant rate (approximately 3.5 frames per second), but a further distance has been traversed in

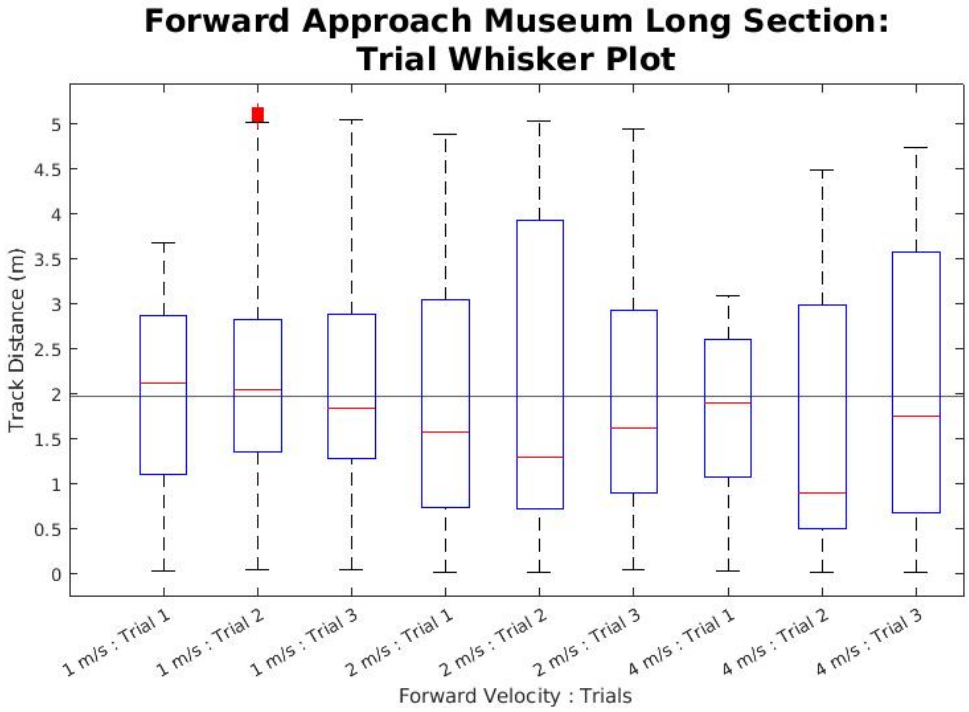


Figure 5.7 Whisker plot of the track distance values of all nine trials in the third forward-facing experiment. Each whisker plot represents the median value as the red line as the center of the box, the 75th percentile as the top of the box, the 25th percentile as the bottom of the box, the maximum and minimum non-outlier values as the end points of the lines, and red dots as the outliers. The horizontal line spanning the whole chart marks the overall average distance value.

between each processed frame. However, the total distance of track traversed does not change in any of the trials, so the time of flight is decreased. Because of this decrease in flight time, there is less time for oscillations in the control systems. A lower number of oscillations in controls means a lower amount of time away from the track, effectively smoothing out the TrackNet error across time. Additionally, this effect may be exagerated by the complexity of the track configuration as these oscillations are most prevalent when following a straight section. An increased speed means that the straight section of this track is followed for a shorter amount of time, decreasing the error from oscillations. This effect can also be seen in decrease in average TrackNet error as forward speed increases. Because the overall change in error

is relatively small, it is more significant that the system remains successful at varying forward velocities. This is clear when looking at Fig. 5.7 as there is no significant variance in the trials as the speed increases.

This study both shows the major weakness of the most commonly utilized track following method currently in the literature, namely vanishing point detection, and also proves the success of a novel control solution to this problem. With this solution implemented, a complete railroad tracking and following system is developed and proven to be capable of autonomously following single railroad tracks of varying configuration.

5.2 Downward-Facing Experimentation

The downward-facing approach went through a similar series of experiments, beginning with initial testing at the shorter University of South Carolina track location, Fig. 5.1. In contrast to the forward-facing approach, this system utilized two separate controllers for yaw and roll from the beginning. After the preliminary controller gain tuning in the lab, the gains were as follows: For the yaw controls, $K_p = 1$ and $K_d = 0.2$; For the roll controls, $K_p = 0.8$ and $K_i = 0.05$.

5.2.1 Experiment 1

The first series of tests consisted of nine trials at the University of South Carolina track location (Fig. 5.1) in which the system proved effective, remaining within the track area at all times and never losing visual sight of the track. The TrackNet system once again demonstrated its effectiveness on real tracks with a forward velocity of one meter per second and the same 3.5 frame per second processing speed. The average TrackNet error was 5.8314% in the directional dimension and 11.8800% in the positional dimension.

Table 5.4 Downward-Facing Experiment 1: TrackNet Errors

Trials:	**Direction Error:**	**Position Error:**
Trial 1	7.3842%	14.0008%
Trial 2	4.3367%	11.0607%
Trial 3	6.5860%	7.3427%
Trial 4	4.2365%	4.9092%
Trial 5	5.3814%	12.7409%
Trial 6	4.8199%	9.1573%
Trial 7	7.1175%	15.7292%
Trial 8	4.9724%	15.0872%
Trial 9	7.6483%	16.8958%
Average:	**5.8314%**	**11.8800%**

Although these values are less than those in the forward-facing results, it is important to recall that these are not necessarily comparable across approaches, as their calculation is based on the track interpretation method utilized. However, these errors are useful when testing if the changes made from this experiment to the next were successful or not. In evaluation of the this data, a consistent steady-state error was identified. To solve this problem, the integral gain was increased to 0.2 and the proportional gain was reduced to 0.75.

5.2.2 Experiment 2

Following this proof of concept experiment, the system was brought to the South Carolina Railroad Museum location (Fig. 5.2) for more thorough testing. At this location, two series of tests were conducted. First, a shorter test beginning with a straight section of track and ending with a sharp turn to the left was utilized to determine if the system was capable of traversing long distances and curved tracks (box labeled "short section" in Fig. 5.2). In this series of six trials the UAV successfully handled the longer distance as well as the curve, performing with average TrackNet errors of 2.6825% and 6.2963% for direction and position respectively as well as an average track position error of 1.9751 meters.

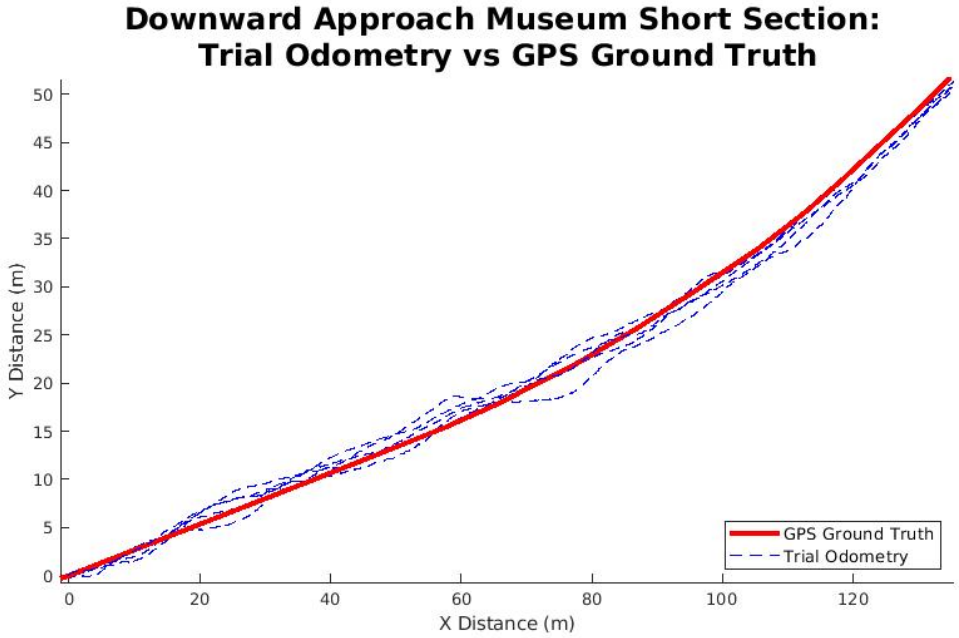


Figure 5.8 Positions of all trials (dotted blue lines) in second downward-facing approach experiment are plotted alongside GPS ground truth track reference (solid red line).

The flight paths of these trials again can be seen in Fig. 5.8, showing the successful following of the track and all of the error values can be seen in Table 5.8.

Table 5.5 Downward-Facing Experiment 2: Track Distance and TrackNet Errors

Trials:	**Track Distance:**	**Direction Error:**	**Position Error:**
Trial 1	1.1112 m	1.2865%	3.9480%
Trial 2	1.7152 m	7.0262%	12.2712%
Trial 3	2.3985 m	1.6444%	4.9538%
Trial 4	2.6861 m	1.3396%	3.5708%
Trial 5	1.8625 m	2.1792%	6.0356%
Trial 6	2.0772 m	2.6195%	6.9981%
Average:	**1.9751 m**	**2.6825%**	**6.2963%**

5.2.3 Experiment 3

Following this success, the system was evaluated along an even longer stretch of track, approximately 300 meters, consisting of two curves (one to the right and one to the

left) and a straight section in between. This test was primarily to ensure the system's effectiveness along longer distances and with more track configurations. Also, this experiment allowed the system to be tested with differing forward velocities, namely one, two, and four meters per second. Overall, the system's performance continued its success, completing the full flight across eight trials, three at one meter per second, three at two meters per second, and two at four meters per second. All of the corresponding errors are detailed in Table 5.6 and the flight paths plotted in Fig. 5.9. The average TrackNet errors were 2.4110% and 7.0089% for direction and position respectively, and the average track position error was 2.0342 meters over all the trials. This further demonstrated the systems ability to maintain a close distance from the track across varying environments. Also, these trials prove the system's effectiveness at increasing forward velocities, but shows a limitation of this method as the error increases with velocity. Specifically, the average track position error increased from 1.7384 meters to 2.2162 meters as from the slowest to the fastest forward velocity. The change in forward velocity has a greater effect on the downward-facing approach due to its limited view of the track. When compared to the forward-facing configuration, the onboard camera can only see a small section of the track. This means that when the forward velocity increases, the system has less time to respond to changing states as it has less information. Here is the only case in these experiments that the processing time becomes the primary limitation because if frames could be processed more quickly this would counteract the decrease in control update speed. In spite of this, the system was still capable of successfully completing this task up to the maximum speed tested, namely four meters per second. It can be seen from Fig. 5.10 that even though there is a slight increase in error as the speed increases, it is still not significant. This seems to imply that the system will remain successful at increasingly high speeds, however this needs further testing to guarantee.

The success of this approach, demonstrates the effectiveness of the downward-

facing approach which is a method scarcely represented and never implemented in the literature. With this a second, fully functional track following system was developed and showed to be successful at autonomously tracking and following railroad tracks.

Table 5.6 Downward-Facing Experiment 3: Track Distance and TrackNet Errors

Forward Vel.:	**Trials:**	**Track Dist.:**	**Dir. Error:**	**Pos. Error:**
1 m/s	Trial 1	1.1105 m	3.9616%	3.1660%
1 m/s	Trial 2	1.7062 m	2.1234%	4.0982%
1 m/s	Trial 3	1.8625 m	1.9072%	6.7553%
1 m/s	**Average:**	**1.7384 m**	**2.6640%**	**4.6732%**
2 m/s	Trial 1	2.3985 m	2.2407%	4.2554%
2 m/s	Trial 2	2.6861 m	1.4399%	3.2353%
2 m/s	Trial 3	2.0772 m	1.7839%	5.1393%
2 m/s	**Average:**	**2.2086 m**	**1.8215%**	**4.2100%**
4 m/s	Trial 1	2.4964 m	1.4577%	4.9364%
4 m/s	Trial 2	1.9359 m	4.3736%	24.4860%
4 m/s	**Average:**	**2.2162 m**	**2.9156%**	**14.711%**
Overall	**Average:**	**2.0342 m**	**2.4110%**	**7.0089%**

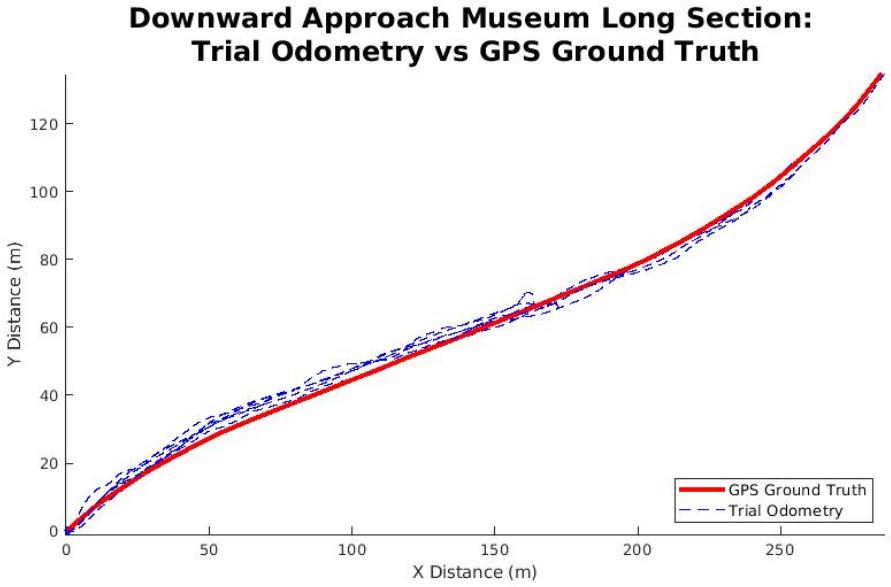


Figure 5.9 Positions of all trials (dotted blue lines) in third downward-facing approach experiment are plotted alongside GPS ground truth track reference (solid red line).

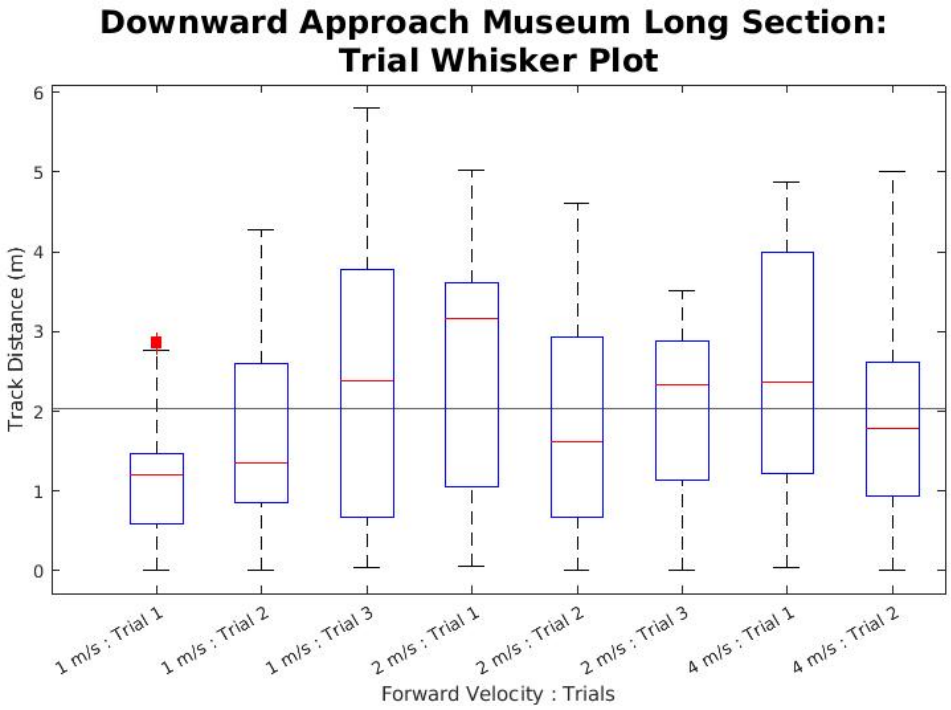


Figure 5.10 Whisker plot of the track distance values of all eight trials in the third downward-facing experiment. Each whisker plot represents the median value as the red line as the center of the box, the 75th percentile as the top of the box, the 25th percentile as the bottom of the box, the maximum and minimum non-outlier values as the end points of the lines, and red dots as the outliers. The horizontal line spanning the whole chart marks the overall average distance value.

Chapter 6

Conclusion

The primary contribution of this work lies in presenting the first thoroughly tested implementation of a UAV system designed for following railroad tracks, leveraging cutting-edge computer vision-based track detection algorithms. The system demonstrates the development of a highly effective track detection algorithm using computer-vision technologies and its integration onboard a customized DJI Matrice 100 UAV. The track detection algorithm, TrackNet, utilizes a compound region-rail detection approach. This process begins with a convolutional neural network, [20], for track region segmentation followed by edge and line detection within this region. These lines are then chained together to estimate the most accurate prediction of the rail lines. Although this system is based on similar approaches in the literature it is novel in a few key ways. Firstly, the structure of TrackNet as a compound region-rail detection algorithm is not necessarily unique as an idea, but is scarcely represented. Additionally, the few examples of such a technique in the literature are rudimentary in the methods used for either stage, particularly for region detection. In contrast, TrackNet utilizes the best of both region and rail detection methods in a way that is yet to be attempted. Secondly, the rail selection process of TrackNet is an original solution to this stage of the process. The concept of creating line chains to account for all possible track configurations provides an increase robustness to the system when compared to other approaches. Finally, the development of trajectory detection as a rail interpretation method alongside vanishing point detection provides the possibility for differing view angles and the comparison of downward and forward approaches.

Such a comparison has not yet been conducted, as the trajectory approach itself is scarcely been attempted.

In addition to the development of TrackNet, two flight control systems were implemented based around the track interpretation methods of TrackNet. These approaches differ in the configuration of the onboard camera: one employing a forward-facing camera configuration and the other utilizing a downward-facing configuration. This division was in effort to validate the efficacy of either approach, to compare their capabilities, and to provide the groundwork required for a full inspection application based on these methods. After initial experimental results, it was determined that both systems would require control of two degrees of freedom, yaw and roll. The yaw controls would ensure the UAV traversed along the correct direction and the roll controls would prevent it from trailing away from the track's position. With these systems in place, both approaches were tested across three series of experiments each.

Although the most prevalent method for controls in the literature was the forward-facing approach utilizing only yaw controls, our implementation of this did not hold in extensive testing. The vanishing point detection that the control system was based on was effective in estimating the direction of the track, but it failed to determine the UAV's proximity to the track. The issue was clear after examining the data, the vanishing point error remained steady even as the track position error increased as seen in Fig. 5.5. Later, it was seen that over longer lengths of track, the UAV would drift away from the track region, but the vanishing point's location would fail to correct for this drift. This issue became particularly problematic when the UAV attempted to traverse a curve in the track. For this reason the aforementioned roll control system was implemented and tested. The roll controller prevented the slow positional drift and corrected whenever it became an issue allowing the successful following of the 300 meter track with an average track position error of 1.9766 meters. This process identified a flaw in the theoretical approach most prevalent in the literature and pro-

vided a solution. This flaw was discovered because the system implemented here is the first of its kind tested live, onboard a UAV, over varying track conditions. Not only did the system prove successful, but was also tested with three forward velocities and proved effective at each.

On the other hand, the downward-facing approach (a minority method in the literature) was developed with both yaw and roll controls from the start, eliminating the potential for a similar issue that its forward-facing counterpart encountered. This system is the first implementation of autonomous track following with a downward-facing camera configuration and the results proved its effectiveness, maintaining an average track distance error of 2.0342 meters across eight trials and three forward velocities. Although this approach performed worse than the forward-facing approach when considering all eight trials, it instead performed better when only considering the one meter per second trials with 1.7384 meters on average. This shows how the system's efficacy decreases as the forward velocity increases. Due to the perspective of the downward-facing camera, it cannot sense sections of track further away from it. This both causes difficulties in controls as well as inspection when the forward speed increases. Because the downward-facing camera cannot sense far ahead, it is inherently limited in the time the controllers have to respond to changing track states. Nevertheless, both approaches proved to maintain successful autonomous track following across 300 meters of track.

With the bigger picture in mind, it is important to evaluate more aspects of each approach than pure system efficacy. In particular, the view of the downward-facing camera is much more predictable when compared to that of the forward-facing camera. The downward-facing view observes the track as a plane parallel to that of the image frame. This assumption makes several inspection parameters much easier to determine. For instance, the area directly above the track is clear as it overlaps with the visible track itself. With the forward-facing perspective, however, some

reconstruction needs to be done to determine what areas in the image are located above the track. Additionally, for the downward-facing perspective no adjustments need to be made to determine the correct size or shape of track defects that are detected. For the forward-facing camera, the angle of the camera as well as the altitude of the UAV will affect how such things are perceived and will require further calculations. Another aspect to be considered is the flight altitude. In order for the downward-facing camera to keep the full track within frame, it had to fly at a higher altitude. This creates a limitation in the system, preventing it from track rails when below a certain altitude. Additionally, this required altitude was outside of the common track kinematic envelope (area which must be kept clear of structures) creating potential safety concerns.

With all of these factors considered, a comparison can be made between the two approaches. There is not, however, a clean victor that is superior in all circumstances, instead it depends on the needs of the application. A table comparing the situations in which each approach is more appropriate can be seen in Table 6.1. In this table there are several factors that may be important for any given application and which approach is more suited to that case. From the testing outlined here the forward-facing approach is likely to be the best option if travel speed, altitude variety, or safety are the aspects of greatest concern. On the other hand, the downward-facing approach is a better choice if following efficacy or ease of inspection are of most importance.

Table 6.1 Forward-Facing Approach and Downward-Facing Approach Comparison

Critical Factor:	**Forward-Facing:**	**Downward-Facing:**
Following Efficacy		X
Travel Speed	X	
Altitude Variety	X	
Safety	X	
Ease of Inspection		X

Moving forward, a more robust system would integrate both a forward-facing camera as well as a downward-facing camera. The utilization of both cameras would allow for the benefits of either option. Additionally, an integration of the control systems would then be possible and would increase the robustness and improve the overall following efficacy. This combination would also assist in inspections types allowing for multiple perspectives to be compared and analyzed to increase accuracy. With both systems online, a vast array of different methods for inspection would be possible. The UAV could prioritize the forward-facing camera for high speed inspection and then slow down when a potential area of interest was detected, increasing its reliance on the downward-facing camera. Additionally, a system could be created where the UAV flies at high altitudes (downward-facing camera) to create a large scale map of the track, then scan this section at a lower height (forward-facing camera) and repeat.

In addition to this major change of approach, there are potential areas for improvement in the system as it currently exists. The most significant relate to further development into more sophisticated control methods. Here, the system only controls two degrees of freedom with independent PD and PI controllers. There are several improvements that could be made to this including: the control of additional degrees of freedom, coupling of the controllers, prediction of future states and consideration of past states.

Overall, this project presents the first successfully tested implementation of autonomous vision-based track detection and following live and onboard a UAV.

Bibliography

[1] Jeremy Atack and Robert A. Margo. "The impact of access to rail transportation on agricultural improvement: The American Midwest as a test case, 1850â1860". In: *Journal of Transport and Land Use* 4.2 (2011), pp. 5–18. ISSN: 19387849. (Visited on 03/06/2024).

[2] Milan Banić et al. "Intelligent machine vision based railway infrastructure inspection and monitoring using UAV". In: *Facta Universitatis, Series: Mechanical Engineering* 17.3 (2019), pp. 357–364.

[3] Alexandre S. Brandão, Felipe N. Martins, and Higor B. Soneguetti. "A vision-based line following strategy for an autonomous UAV". In: *2015 12th International Conference on Informatics in Control, Automation and Robotics*. Vol. 02. 2015, pp. 314–319.

[4] Anthony Clerc. "Tracking of railroads for autonomous guidance of UAVs: using Vanishing Point detection". PhD thesis. KTH Royal Institute of Technology, 2018.

[5] Dave Donaldson. "Railroads of the Raj: Estimating the Impact of Transportation Infrastructure". In: *American Economic Review* 108.4-5 (2018), pp. 899–934. DOI: 10.1257/aer.20101199.

[6] Vigile Marie Fabella and Sonja Szymczak. "Resilience of Railway Transport to Four Types of Natural Hazards: An Analysis of Daily Train Volumes". In: *Infrastructures* 6.12 (2021). ISSN: 2412-3811.

[7] Lance R Grenzeback, Andrew T Lukman, and Cambridge Systematics. *Case study of the transportation sector's response to and recovery from hurricane's Katrina and Rita*. Transportation Research Board, 2008.

[8] Feng Guo, Yu Qian, and Yuefeng Shi. "Real-time railroad track components inspection based on the improved YOLOv4 framework". In: *Automation in Construction* 125 (2021), p. 103596. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2021.103596.

[9] Emre Güçlü, İlhan Aydın, and Erhan Akın. "Fuzzy PID Based Autonomous UAV Design for Railway Tracking". In: *2021 International Conference on Information Technology (ICIT)*. 2021, pp. 456–461. DOI: 10.1109/ICIT52682.2021.9491687.

[10] Allam Shehata Hassanein et al. *A Survey on Hough Transform, Theory, Techniques and Applications*. 2015. arXiv: 1502.02160 [cs.CV].

[11] Mehmet Karakose et al. "A new computer vision based method for rail track detection and fault diagnosis in railways". In: *International Journal of Mechanical Engineering and Robotics Research* 6.1 (2017), pp. 22–17.

[12] B. Le Saux et al. "Railway Detection: From Filtering to Segmentation Networks". In: *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. 2018, pp. 4819–4822. DOI: 10.1109/IGARSS.2018.8517865.

[13] Keith Lewandowski and Nikolaos Vitzilaios. "UAV-Based Railroad Line Detection". In: *Joint Rail Conference (JRC)*. 2024.

[14] Xiang Liu, M. Rapik Saat, and Christopher P. L. Barkan. "Analysis of Causes of Major Train Derailment and Their Effect on Accident Rates". In: *Transportation Research Record* 2289.1 (2012), pp. 154–163. DOI: 10.3141/2289-20. eprint: https://doi.org/10.3141/2289-20.

[15] Koppány Máthé and Lucian Buşoniu. "Vision and Control for UAVs: A Survey of General Methods and of Inexpensive Platforms for Infrastructure Inspection". In: *Sensors* 15.7 (2015), pp. 14887–14916. ISSN: 1424-8220.

[16] Andrei I. Purica, Béatrice Pesquet-Popescu, and Frédéric Dufaux. "A railroad detection algorithm for infrastructure surveillance using enduring airborne systems". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2017), pp. 2187–2191.

[17] Előd Páli et al. "Railway track following with the AR.Drone using vanishing point detection". In: *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*. 2014, pp. 1–6. DOI: 10.1109/AQTR.2014.6857870.

[18] Zhiquan Qi, Yingjie Tian, and Yong Shi. "Efficient railway tracks detection and turnouts recognition method using HOG features". In: *Neural Computing and Applications* 23.1 (July 2013). ISSN: 1433-3058. DOI: 10.1007/s00521-012-0846-0.

[19] Yihao Ren et al. "Review of Emerging Technologies and Issues in Rail and Track Inspection for Local Lines in the United States". In: *Journal of Trans-*

portation Engineering, Part A: Systems 147.10 (2021), p. 04021062. DOI: 10.1061/JTEPBS.0000567.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597.

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597.

[22] Arun Kumar Singh et al. "Vision based rail track extraction and monitoring through drone imagery". In: *ICT Express* 5.4 (2019), pp. 250–255. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2017.11.010.

[23] Evan McLean Smith. "A collection of computer vision algorithms capable of detecting linear infrastructure for the purpose of UAV control". PhD thesis. Virginia Tech, 2016.

[24] Zhu Teng, Feng Liu, and Baopeng Zhang. "Visual railway detection by superpixel based intracellular decisions". In: *Multimedia Tools and Applications* 75 (2015), pp. 2473 –2486.

[25] Antonio Torralba, Bryan C. Russell, and Jenny Yuen. "LabelMe: Online Image Annotation and Applications". In: *Proceedings of the IEEE* 98.8 (2010), pp. 1467–1484. DOI: 10.1109/JPROC.2010.2050290.

[26] Yin Wang et al. "RailNet: A Segmentation Network for Railroad Detection". In: *IEEE Access* 7 (2019), pp. 143772–143779. DOI: 10.1109/ACCESS.2019.2945633.

[27] Yunpeng Wu et al. "Hybrid deep learning architecture for rail surface segmentation and surface defect detection". In: *Computer-Aided Civil and Infrastructure Engineering* 37 (June 2021). DOI: 10.1111/mice.12710.

[28] Oliver Zendel et al. "RailSem19: A Dataset for Semantic Rail Scene Understanding". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2019), pp. 1221–1229. URL: https://api.semanticscholar.org/CorpusID:198166233.

[29] Weihua Zhu et al. "Seismic risk assessment of the railway network of China's Mainland". In: *International Journal of Disaster Risk Science* 11 (2020), pp. 452–465.