Multi-modal Learning Pipeline for Smooth Georeferenced Tracking from an Uncalibrated Monocular Camera

Anonymous Author(s)
Affiliation
Address
email

Abstract

Automatic generation of georeferenced trajectories for vehicular traffic is critical for situational awareness and monitoring for both manned and unmanned platforms, irrespective of their level of autonomy. The goal is to estimate the world coordinates of one or more vehicles from a sequence of images with known camera world coordinates. Traditional depth estimation techniques rely on cues from feature correspondences of multiple viewpoints or knowledge of the camera parameters based on extensive camera calibration. However, camera calibration is a complex and challenging error-prone task. Also, these camera parameters are not guaranteed to be estimable to desired accuracy. This paper presents a lightweight technique to generate georeferenced tracks of vehicular traffic from monocular camera images using a supervised multi-layer model trained on a sparse dataset. The developed pipeline chains well-known highly efficient detection algorithm with novel multi-modal inference learning algorithm, physics-driven interpolation algorithm and an error evaluation process. The novelty of this inference model is the combination of a highly generalizable layer with an extra precision layer, resulting in a highly accurate process with minimal performance degradation across testing and validation datasets. The developed pipeline was tested using real-world maritime traffic images generated from a generic low-cost webcam with, unknown parameters, mounted on a vessel with known GPS coordinates. The whole pipeline was implemented on an NVIDIA Jetson TX2, an embedded system-on-module (SoM) with dual-core NVIDIA Denver2 + quad-core ARM Cortex-A57, 8GB RAM and integrated 256-core Pascal GPU. The results show that the pipeline produces very accurate world coordinate estimates of unseen data using minimal features with unknown camera parameters. The results also show that proposed inference technique is robust to detection uncertainties. [1]

1 Introduction

Accurate observation and understanding of surroundings are crucial for navigation, whether for humans, animals, or computers [1]. This understanding is essential for tasks like coordination, path planning, obstacle avoidance [2], prediction, and localization [3]. For an autonomous vehicle, an understanding of the surrounding involves estimating the location of objects in the surrounding [4], predicting their movement, and then planning a path to achieve its desired goals [3]. In the case

[1] The initial stages of the pipeline presented in this paper won the 2nd place in the US Navy's *AI Tracks at Sea* 2020 challenge. The current version of the software is open source at `https://github.com/fsudssAI/AI-track-at-sea`. [2].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

of traffic observation, perception can be used to monitor and coordinate the movement of multiple vehicles [5].

To accomplish this, eyes and other senses are used by humans to plan their movement, avoid obstacles, predict where other moving objects will be at some time in the immediate future while engaging in any form of movement like walking, running, or driving. Computers also need to be able to perceive their surroundings and this is accomplished using sensors. Some sensors currently used to perceive surroundings are LIDAR scanners [6], RADAR, ultrasonic sensors, and cameras. These sensors can be used independently or together through a process called sensor fusion for coordination – path planning, obstacle avoidance, prediction, localization, and vessel traffic monitoring. However, each of these sensors has its benefits and drawbacks. LIDAR, for example, can accurately localize objects with great resolution but is expensive and heavy, RADAR can accurately sense objects but has minimal resolution and cannot be used to distinguish objects, and ultrasonic sensors have very limited range. On the other hand, cameras represent a cost-effective alternative to these sensors, distinguish objects, have color information, and provide information-dense data similar to the human eye. However, cameras lack depth information due to the loss of information from the transformation from three dimensions to two in generating images. This ambiguity makes it difficult for computers to understand their surroundings using a single camera alone accurately. However, due to its versatility, low cost, small form factor, and use in a broad range of applications, depth estimation using a camera is a viable solution.

Much work has gone into depth estimation – recovering the depth information – from camera images. **Homography-based methods** recover 3D data from camera calibration [7]–[9], in which a homography or more generally, transformation [10] matrix is estimated from the camera parameters – camera intrinsics and extrinsics. The camera intrinsics use the camera specifications – focal point, optical center – to map pixel coordinates in the image frame to coordinates in the camera frame. The camera extrinsics takes cues from the camera's orientation to transform the coordinates from the camera frame to the world frame. The drawback to this is that the exact camera specifications and orientation are assumed to be known. **Geometry-based methods** attempt to recover 3D structures from multiple images based on geometric constraints. *Stereo vision matching* [7], [11], recovers the 3D structure [12] of a scene using multiple cameras and takes advantage of the multiple viewpoints. Stereo vision matching is similar to the way humans perceive depth. However, Stereo Vision matching assumes the cameras have been calibrated in advance, i.e., the transformation between the cameras is known precisely. Therefore the scale information must be included in the depth estimation process. *Structure from motion* (SfM) [13], [14] estimates 3D structures of a scene from a sequence of 2D images. This method has been applied in a variety of tasks, from vSLAM [15], 3D reconstruction [16], velocity estimation [17]–[19]. Depth is estimated from sparse features using known geometric constraints and feature correspondences between the images. However, this is only accurate up to an unknown scale factor [9], due to the scale ambiguity of a monocular camera. Furthermore, this method is not robust and requires high-quality image sequences and exact feature matching between images. Another well-investigated method is through the use of *deep neural networks* [19]–[22], which benefit from the recent improvements in computational power. Deep neural networks have shown success in tasks like image classification [23] and object detection [24] among others. This involves training a model to predict the depth of pixels from ground truth data obtained from other sensors like LIDAR or GPS coordinates in an end-to-end fashion [25].

Although the above methods estimate the depth of sparse pixels in world coordinates, they either depend on accurate apriori knowledge of the camera parameters, multiple cameras, image sequences, very high-quality images [16], or high computational demand in the case of deep neural networks. Generating georeferenced tracks from a single monocular camera and image is still significantly challenging. A method is proposed in this paper to estimate the 3D coordinates from camera frames using GPS coordinates as the training data. In this paper, this challenge is tackled as a problem of monitoring maritime vehicular traffic. Traditional methods to track maritime traffic rely on RADAR which poses a significant risk in the presence of adversaries. Therefore, a monocular camera represents a safe and accurate alternative. A three-stage pipeline is created to generate 2D bounding boxes around detect objects of interest in the image, transform the centroids of the bounding boxes to 3D world coordinates, and finally, generate a smooth trajectory from a sequence of 3D tracks.

The contributions of this paper are three-fold: first, a model is trained to transform the 2D image coordinates of a vehicle to 2D world coordinates (depth estimation). Second, a subsystem to increase the accuracy of the model by integrating the kinematics of motion of tracked vehicles is created by

smoothing the predicted world coordinates and removing outliers. Finally, an evaluation routine is implemented to compare the ground truth and predictions.

The remainder of the paper is organized as follows: In section **??**, necessary notations are clarified for the subsequent development. In section 3, we discuss the difficulties on automation generation of georeferenced trajectories and state the contribution of this paper. In section 4, we describe the proposed method in detail, which contains object detection, inference, trajectory interpolation and trajectory evaluation. In section 5, training and testing results are presented to show good performance of the proposed method, and software implementation is also introduced. Conclusion remarks follow in section 6.

2 Notation

The following notations and definitions are used throughout the whole paper: $\mathbb{R}, \mathbb{R}^n, \mathbb{R}^{n\times m}$ denote the space of real numbers, real vectors of length n and real matrices of n rows and m columns respectively. $\mathbb{R}_+$ denotes the space of positive real numbers. Normal-face lower-case letters $(e.g.\ x \in \mathbb{R})$ are used to represent real scalars, bold-face lower-case letters $(e.g.\ \mathbf{x} \in \mathbb{R}^n)$ represent vectors, while normal-face upper-case letters $(e.g.\ X \in \mathbb{R}^{n\times m})$ represent matrices. $X^\top$ denotes the transpose of matrix X. Let $\mathbb{S}^{n\times n} \subset \mathbb{R}^{n\times n}$ denote a set of symmetric matrices, such that $X \in \mathbb{S}$ implies $X = X^\top$. $\ominus$ denotes the inverse motion composition operator. Given a matrix $A \in \mathbb{R}^{m\times n}$, a pseudoinverse of A is denoted by A^+ satisfying Moore-Penrose conditions [26].

3 Problem Statement

This paper proposes a lightweight solution to georeferenced tacking of vehicular traffic based on uncalibrated monocular camera. Firstly, the detection-based approach is more economic compared to sensor-based, such as LIDAR, RADAR or GPS, approaches, and the detection technique has gained big progresses by employing different learning-based approaches. Secondly, we assume the camera's parameters are unknown, and only single camera is used to capture the vehicular traffic. Thus, the traditional depth estimation methods, discussed in section 1, are excluded. However, by observing the underlying relationship between image coordinates and world coordinates, the inference task of georefrenced trajectory can be seen as a data-driven regression problem. In this paper, we design a lightweight kernel regression method with residual correction term which gives a surprised inference performance. Moreover, the trajectory smoothing and evaluation are also included to generate a smooth tracking path and evaluate the tracking precision respectively. The structure of the whole solution is shown in Figure. 1.

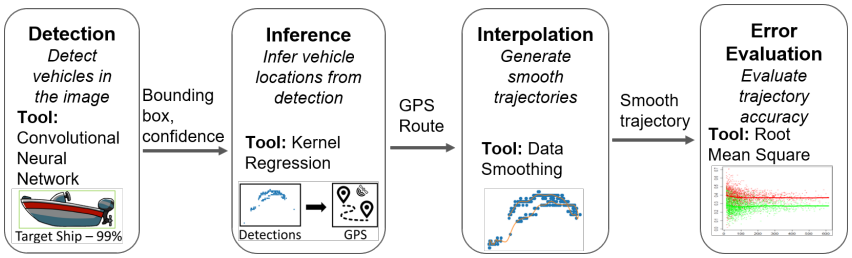

Figure 1: A schematic of the information flow between subsystems of the pipeline.

The first subsystem in the pipeline is the detection subsystem which takes as input a video from a single camera and generates bounding boxes around the target ship of interest. This subsystem does not only detect the ship, but also track them in subsequent image frames. This can be particularly challenging since the target ship can be occluded by other objects in subsequent frames, it can leave the camera field of view (FOV) and is also similar in appearance to many other shipping vessels that enter the FOV. Therefore, this subsystem should be able to accurately localize vehicles in the image regardless of occlusions or other obstacles.

A subsequent challenge is the inference of corresponding bounding box image coordinates in GPS coordinates. This challenge is exacerbated by the absence of the camera calibration parameters

from the monocular videos, thereby rendering any of the previously highlighted homography-based or geometric-based methods from section 1 useless. Therefore, the inference subsystem in the pipeline must take as input the results from the detection subsystem, handle the ambiguity of depth estimation from a single uncalibrated camera and output the transformation from image coordinates to normalized GPS coordinates.

The next subsystem takes a sequence of GPS coordinates from the Inference subsystem and generates smooth trajectories from them. However, the detections obtained from the Detection subsystem are very noisy, due to false detections and high execution rate (multiple times a second). The resulting GPS coordinates from the Inference subsystem are therefore noisy. The interpolation module must then filter the outliers (false detections and misses) and the noisy coordinates to a smooth trajectory while also taking into account realistic motion of vehicles.

Finally, to test the efficacy of the pipeline, a survey of various error evaluation techniques was done. The main challenges here arise from the time correspondence between the ground truth data and the predictions as well as the projection from GPS coordinates to (X, Y) coordinates for evaluation. The time correspondence problem is due to the variable sample time of the GPS sensor used to acquire the ground truth data and the sample time of the pipeline. The projection challenge is necessary because evaluating the errors directly with GPS coordinates is ill-posed because a small error in GPS coordinates translates to meters.

4 Solution Approach

In this section, the end-to-end pipeline in Figure. 1 is explored in details. The detection subsystem generates bounding boxes around each vehicle with corresponding detection confidence. The inference subsystem transforms the detection results from pixel coordinates to world coordinates, and the interpolation subsystem then generates a smooth trajectory from the transformed points and filter outlier detections. Finally, an error evaluation subsystem calculates the deviation of the predictions from the ground truth to test the accuracy of the entire pipeline.

4.1 Object Detection

Accurate identification and tracking of the target vehicle in the image stream are crucial for predicting a reliable trajectory. Object detection and object tracking are well-researched problems with state-of-the-art models released such as YOLOv3, YOLOv4, EfficientDet, and Faster R-CNN. The performance of these existing state-of-the-art object detection models was evaluated on a custom dataset using the mean average precision [27] (mAP @ 0.5 IoU) [3], training time, inference time, and model size as metrics.

EfficientDet is an object detection model built with an EfficientNet CNN backbone with several optimizations, such as the use of a BiFPN, and a compound scaling method that uniformly scales the resolution, depth, and width for all backbones, feature networks, and box/class prediction networks at the same time [28]. **Faster-RCNN** is a two-stage object detector featuring a Regional Proposal Network (RPN) for identifying candidate regions and a separate network that uses these candidates for the final detection. **You Only Look Once version 3 (YOLOv3)**[29], uses a variation of Darknet[30] for its CNN backbone by stacking a 53 layer detection head on the already 53 convolutional layers of Darknet. **YOLOv4** is the successor to the YOLOv3 architechture with CSPDarknet53, based on DenseNet, as the CNN backbone. YOLOv4 was designed to remove the computational bottlenecks of DenseNet and improve learning by passing on an unedited version of the feature map. CSPDarknet53 has 29 convolutional layers with 20.6 million parameters [31], approximately three times fewer parameters than its predecessor, YOLOv3. YOLOv4 features a "Bag of Freebies" - which are mainly data augmentation techniques to improve model performance without additional inference time. YOLOv4 also features a "Bag of Specials" which are strategies that offer a significant increase in performance for a marginal increase in inference time.

The training and testing datasets comprised of raw unlabeled videos captured on a generic webcam. The training data was augmented by rotation, brightness, exposure, blur, and noise to make the model robust to variations in the picture and improve generalizability of the model.

[3] Cartucho mAP GitHub repository

4.2 Inference

The next step is to obtain the corresponding georeferenced trajectory with the bounding boxes of the vehicles and the detection confidence obtained. However, due to the lack of camera parameters and the single source of recording images, the traditional depth estimation methods cannot be used as described in Section 1. Therefore, a data-driven approach was developed to learn a coordinate transformation function $f : \mathbb{R}^2 \to \mathbb{R}^2$ which captures the underlying relationship between the normalized image coordinates and the corresponding normalized world distance between the vehicles and the camera. Specifically, in this section, a two-layer regression model is developed; The first layer uses a lightweight quadratic kernel regression model to capture most of the learning and improve generalizability. The second layer then uses a higher-order learning model like Deep Neural Network (DNN) or Gaussian Process Regression (GPR) to learn higher frequency features from the residual of the quadratic fit to improve the accuracy of the overall model.

Formally, the novel coordinate transformation problem is formulated as follows: given training dataset $\{X, Y\}$, where $X = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(m)}\right\}$, $Y = \left\{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \cdots, \mathbf{y}^{(m)}\right\}$, and $\mathbf{x}^{(i)} \in \mathbb{R}^2, \mathbf{y}^{(i)} \in \mathbb{R}^2$, for $i = 1, 2, \cdots, m$, learn the transformation function $f : \mathbb{R}^2 \to \mathbb{R}^2$ for the unknown camera. The normalized image coordinates $\mathbf{x}$ contains x-coordinate $\mathbf{x}_1$ and y-coordinate $\mathbf{y}_1$ with respect to the origin at left top of images. The normalized GPS distance vector $\mathbf{y}$ contains the latitude distance $\mathbf{y}_1$ and the longitude distance $\mathbf{y}_2$. The constitutive model for each component is given by:

$$y = f_q(\mathbf{x}) + f_r(\mathbf{x}), \tag{1}$$

where $f_q : \mathbb{R}^2 \mapsto \mathbb{R}$ is the main quadratic fit component and $f_r : \mathbb{R}^2 \mapsto \mathbb{R}$ is the lower layer residual component. The quadratic model is given by

$$f_q(\mathbf{x}) \triangleq \mathbf{x}^\top H \mathbf{x} + \mathbf{f}^\top \mathbf{x} + r, \tag{2}$$

where, $\mathbf{x} \in \mathbb{R}^2$, $y \in \mathbf{R}$. The goal is to find the parameters $H \in \mathbb{S}^{2\times 2}$, $\mathbf{f} \in \mathbb{R}^2$, $r \in \mathbb{R}$ such that the loss function

$$J_q(H, f, r) = \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - f(\mathbf{x}^{(i)}) \right)^2 \tag{3}$$

is minimized.

Generally, for $\mathbf{x} \in \mathbb{R}^n$ in the following solution scheme, so the solution should output $H \in \mathbb{S}^{n\times n}$, $\mathbf{f} \in \mathbb{R}^n$, $r \in \mathbb{R}$.

The quadratic model in (2) can be rewritten as

$$f(\mathbf{x}) = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top Q \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \quad \text{with} \quad Q = \begin{bmatrix} H & \frac{1}{2}\mathbf{f} \\ \frac{1}{2}\mathbf{f}^\top & r \end{bmatrix}.$$

where $Q \in \mathbb{S}^{(n+1)\times(n+1)}$. Let $Z = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \in \mathbb{S}^{(n+1)\times(n+1)}$, then

$$f(x) = \mathsf{trace}(QZ) =< Q, Z >= \mathbf{z}^\top \mathbf{q}$$

where, $\mathbf{z} = \mathsf{vec}(Z), \mathbf{q} = \mathsf{vec}(Q)$ are vectorizations of matrix Z, Q respectively.

Next, the loss function (3) becomes

$$\begin{aligned} J_q(\mathbf{q}) &= \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - \mathbf{z}^{(i)\top} \mathbf{q} \right)^2 \\ &= \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)2} - 2y^{(i)} (\mathbf{z}^{(i)\top} \mathbf{q}) + \mathbf{q}^\top \mathbf{z}^{(i)} \mathbf{z}^{(i)\top} \mathbf{q} \right) \\ &= \frac{1}{2} \sum_{i=1}^{N} y^{(i)2} + \left(\sum_{i=1}^{N} \mathbf{y}^{(i)} \mathbf{z}^{(i)} \right)^\top \mathbf{q} + \frac{1}{2} \mathbf{q}^\top \left(\sum_{i=1}^{N} \mathbf{z}^{(i)} \mathbf{z}^{(i)\top} \right) \mathbf{q} \end{aligned}$$

Taking derivative and setting to zero yields

$$\left(\sum_{i=1}^{N} \mathbf{z}^{(i)} \mathbf{z}^{(i)\top} \right) \mathbf{q} = \sum_{i=1}^{N} \mathbf{y}^{(i)} \mathbf{z}^{(i)}$$

Let $\sum_{i=1}^{N} \mathbf{z}^{(i)}\mathbf{z}^{(i)\top}$ admit the singular value decomposition such that

$$\sum_{i=1}^{N} \mathbf{z}^{(i)}\mathbf{z}^{(i)\top} = U\Sigma V^{\top},$$

then

$$\mathbf{q} = V\Sigma^{+}U^{\top}\sum_{i=1}^{N} \mathbf{y}^{(i)}\mathbf{z}^{(i)}.$$

Finally, the dataset $\{X, Y - f_q(X)\}$ is used to train the residual model by minimizing the residual loss function:

$$J_r(H, f, r) = \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \mathbf{z}^{(i)\top}\mathbf{q} - f_r(\mathbf{x}^{(i)}; \mathbf{q}_r)\right)^2 \tag{4}$$

with respect to the parameter vector $\mathbf{q}_r$. The optimization problem above is solved using a back-propagation if f_r is a DNN and Bayesian inferencing if it is a GPR model. Consequently, the inference algorithm is given below in Algorithm 1.

Algorithm 1: Two-layer training algorithm with Quadratic fit and higher order residual model.

Initialization: $A = 0_{m\times m}, \mathbf{b} = 0_{m\times 1}$, where $m = \frac{1}{2}(n+1)(n+2)$.

1. Iterate on the training data:
   for $i = 1$ to N :

$$Z_i = \begin{bmatrix}\mathbf{x}_i \\ 1\end{bmatrix}\begin{bmatrix}\mathbf{x}_i \\ 1\end{bmatrix}^{\top}, \quad \mathbf{z}_i = vec(Z_i)$$

$$A = A + \mathbf{z}_i\mathbf{z}_i^{\top}, \quad b = b + y_i\mathbf{z}_i$$

   end for
2. Solve for quadratic weights in the feature space: $\mathbf{q} = A/b$
3. Solve for optimal residual weights $\mathbf{q}_r^*$ by minimizing (4).

Output:

1. Compute feature vector:

$$Z = \begin{bmatrix}\mathbf{x} \\ 1\end{bmatrix}\begin{bmatrix}\mathbf{x} \\ 1\end{bmatrix}^{\top}, \quad \mathbf{z} = vec(z)$$

2. output coordinate inference:

$$y = \mathbf{z}^{\top}\mathbf{q} + f_r(\mathbf{x}; \mathbf{q}_r^*).$$

4.3 Trajectory Interpolation and Extrapolation

Given the timestamp and resulting predictions of georeferenced tracks from the inference subsystem, the tracks are interpolated and extrapolated to obtain a smooth trajectory. This is done in three steps: moving-average weighted smoothing and outlier-rejection, spline interpolation, and extrapolation based on first-order kinematics.

The dataset used for the trajectory estimation consists of the frame number, the latitude and longitude distance from the camera (obtained from the inference subsystem), and detection confidence (obtained from the detection subsystem). Due to noise and false detections, the low confidence detection results represent outliers in the trajectory dataset, which leads to an adverse impact on the inference subsystem performance. Thus, the first step is to reject outliers and smooth the noisy trajectory points. To accomplish this, a moving average filter [32] is used to smooth the short-term variations among the data and highlight other important features of data, such as trends of the trajectory. The procedure is as follows: choose a fixed size of the rolling window, then calculate the mean of the data points in the moving window and update the smoothing data points by admitting slight deviations. Furthermore,

the detection confidence is included as weights to incorporate the trajectory trends into the moving average filter and implemented the weighted-moving average method by using *pandas.series.rolling*[4].

Next, the missing data points are compensated for in order to produce a continuous and smooth trajectory. Spline interpolation [33] is used in this paper, which fits low-degrees polynomials to each small subsets of data points instead of fitting a single high-degree polynomial for the whole dataset. Therefore, the superiority of spline interpolation on real-world trajectory compared to other interpolation methods, such as Kernel Regression [34], Kalman Filter [35], is that it is difficult to encapsulate the trajectory by a single function, model, or curve. The real-world trajectory might admit a vastly different class of curves [36], and by fitting curves locally, the underlying complex curves might be realized.

The final step, extrapolation, is done by considering the underlying kinematics of vehicles using a first-order kinematic model. Combining these three steps performs better than individually due to the trajectories' complexity and underlying physical meaning.

4.4 Trajectory Evaluation

In this subsection, an evaluation process is introduced to quantify the accuracy of the proposed algorithm. Evaluating a trajectory entails summarizing the high-dimensional information encoded in the whole trajectory into a single, concise metric. Also, due to the high-frequency nature of the GPS sensor used to acquire the ground truth, a time correspondence between the ground truth and prediction tracks must be made for the duration of the video. Understandably, this makes formulating a quantitative measure to compare the estimated trajectory against the ground truth a nontrivial task. A naive implementation of an absolute metric such as mean-square error is

$$\varepsilon(x_{1:T}) = \frac{1}{T}\sum_{t=1}^{T}\left(x_t \ominus x_t^*\right)^2,$$

where ε denotes the evaluated error and $\ominus$, the inverse motion composition operator, is sub-optimal [37]. A metric was proposed in [38] that does not rely on a global reference frame and instead computes the accuracy based on the relative relations between the poses:

$$\varepsilon(x_{1:T}, \Delta) = \frac{1}{T}\sum_{t=1}^{T}\left(\left(x_{t+\Delta}^* \ominus x_t^*\right) \ominus \left(x_{t+\Delta} \ominus x_t\right)\right)^2.$$

However, several variants of the Relative Pose Error (RPE) have also been proposed in the literature, with some advocating for using the RMS formulation instead of the mean-squared formulation given above, while others have extended the relative pose error to be a function of the sub-trajectories[39]. Since the trajectory returned by the pipeline only has a translational component, which is expressed in the metric space, the metric representation was favored over the geographic coordinates to ensure that the errors are correctly scaled and keep the numerical inaccuracies to a minimum. Also, the estimated trajectory and ground truth trajectory are in the same frame, and hence no transformation is required at the time of evaluation. The two trajectories might, however, not be synchronized, but this is handled by the python library used for trajectory evaluation [40].

5 Results

In order to evaluate the accuracy of the entire pipeline, the performance of each subsystem is evaluated individually. The first two subsystems of the pipeline developed (Detection and Inference) were tested on an unseen 3 minute video with 3600 frames for which the ground truth GPS coordinates were known per frame with a 10 meter margin of error. The results of the predicted tracks (with and without the Interpolation subsystem) were compared using the error evaluation subsystem and then evaluated the performance of the pipeline as a whole.

5.1 Performance of Object Detection Models

In order to train the surveyed object detection networks, approximately two minutes of the training videos were decomposed into individual frames and bounding boxes were hand-labeled in each

[4]https://pandas.pydata.org/docs/reference/api/pandas.Series.rolling.html

frame. Two separate training datasets were then subsequently created with the first dataset containing 2000 training samples with no data augmentation. The second dataset contained 2000 training samples with intentionally applied data augmentations. These augmentations included added rotation, brightness, exposure, blue and noise applied to the original training samples. The below results are the performance of all the models on the test set. For YOLOv4, the effect of data augmentation is analyzed on training performance. YOLOv4 adds data augmentation in its architecture. In addition we applied our added data augmentation through Roboflow. To analyze these effects, training is analyzed for four models with both YOLO and added augmentation, only YOLO augmentation, only added augmentation and no added augmentation. This can be seen in Figure. 2.

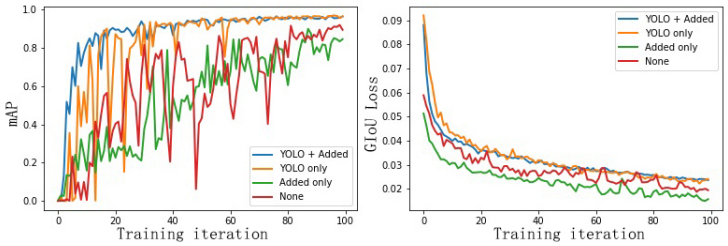

Figure 2: A Comparison of augmentation testing results for different detection methods. (*left*: mAP @ 0.5 IoU augmentation test, *right*: $GIoU$ loss augmentation test. YOLO + Added: , YOLO only: , Added only: , None: .)

In order to compare the results across all of the surveyed object detection models, the results for mAP @ 0.5 IoU, training time, averaged inference time on the test set and model size were recorded. For all models, an increase in precision was observed when training on the augmented dataset. For that reason, all the surveyed models that are compared were trained on the augmented dataset. For models with internally provided data augmentation, they were used on top of the already added augmentation. The comparison can be seen in Table 1.

Comparison of Object Detection Models				
Model	Training Time	Inference Time (Single Frame)	Parameters	mAP @ 0.5 IoU
YOLOv3	3 hr 35 min	50 ms	65.2×10^6	0.83
EfficientDet-D0	2 hr 24 min	53 ms	3.9×10^6	0.82
YOLOv4	3 hr 1 min	40 ms	20.6×10^6	0.89
Faster-RCNN	4 hr 25 min	155 ms	27.6×10^6	0.88

Table 1: Comparison of Object Detection Models Trained on Augmented Dataset

Videos	Metrics	NN	GPR	QKR	QKR + NN	QKR + GPR
V1	MSE	695.30	90.76	67.84	64.20	6.76
	RMSE	26.37	9.53	8.24	8.01	2.60
	Relative MSE	0.10	0.09	0.07	0.07	0.05
	Relative RMSE	0.32	0.31	0.26	0.26	0.23
V2	MSE	3297.59	12734.71	1279.32	2177.81	1215.99
	RMSE	57.42	112.85	35.77	46.67	34.87
	Relative MSE	0.16	0.24	42.04	44.14	0.12
	Relative RMSE	0.40	0.49	6.48	6.64	0.35
V3	MSE	4343.69	87.24	452.57	32.84	40.48
	RMSE	65.91	9.34	21.27	18.246	6.36
	Relative MSE	0.28	0.26	22.03	9.55	2.49
	Relative RMSE	0.53	0.51	4.69	3.09	1.58

Table 2: Table showing the results of each inference method (with interpolation) for 3 videos of the test dataset. The smaller the errors, the better the performance.

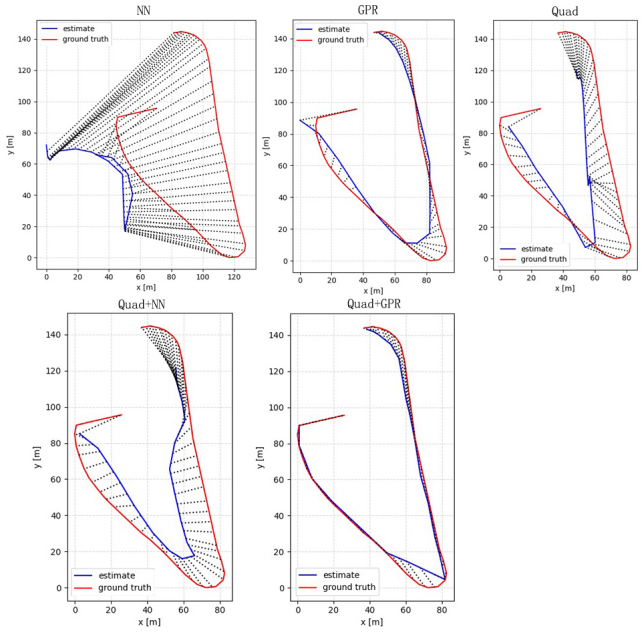

Figure 3: A comparison of map plots generated by the proposed software with five different underlying inference models on test video 3. (NN: neural network, GPR: Gaussian model regression, Quad: kernel regression with quadratic kernel. Red is the ground truth of trajectory, blue is the predicted trajectory.)

5.2 Overall Results

The final output of the pipeline, seen in Figure 3, shows the inference and interpolation results compared to the ground truth data. The Quadratic Kernel Regression (QKR) inference subsystem has errors as shown in 2 and including the interpolation subsystem improved the accuracy by 56.45%. The entire pipeline was fully implemented in python using OpenCV, Tensorflow, UTM, and scikit-learn and run on an Nvidia Jetson TX2 for about 2 minutes 15 seconds. The detection subsystem ran at an average 15 FPS on a GPU with the inference executing at more than 100 hz on a CPU while the interpolation runs at about 10 hz on CPU. The code is available for immediate testing using the docker container located at this Docker repository. The bleeding edge source code is open source and available at this GitHub repository.

6 Conclusion

In this paper, the benefits of using a camera for automatic generation of georeferenced tracks for objects of interest as well as the difficulties in achieving this using a single uncalibrated camera were reviewed. A lightweight end-to-end pipeline comprised of detection, inference, interpolation and error evaluation subsystems was developed to solve this with details for each layer tested on an embedded system. The accuracy of each subsystem in the pipeline was shown; for the detection, YOLOv4 gave an overall mAP score of 0.89, the multimodal inference subsystem, which combines an accurate generalizable layer with a high frequency residual correction, was able to directly regress the georeferenced tracks of the detections and achieved an error of about 2.60 on one video evaluated using the Root Mean Square Error metric discussed in Subsection 4.4. Furthermore, we were able to improve the inference accuracy by taking into account the physics of the objects of interest together with some domain knowledge by generating a smooth trajectory. Future work will address multi-object tracking as well as simultaneous prediction and tracking. The pipeline will then be applied to other fields, such as self-driving cars, aerial vehicles, as well as low cost indoor localization.

7 Appendix

7.1 Vectorization

Definition 7.1 (Verctorization for Symmetric Matrix)**.** *Given a symmetric matrix $X \in \mathbb{S}^{n\times n}$ such that*

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ & x_{22} & x_{23} & \cdots & x_{2n} \\ & & x_{33} & \cdots & x_{3n} \\ & & & & \vdots \\ & & & & x_{nn} \end{bmatrix}, \tag{5}$$

then, by stacking the elements of X diagonal-wise, the vectorization of X is represented by $\mathbf{x} \in \mathbb{R}^{\frac{1}{2}n(n+1)}$ such that

$$\mathbf{x} \triangleq \mathit{vec}(X) = \begin{bmatrix} x_{11} \\ x_{22} \\ \vdots \\ x_{nn} \\ \sqrt{2}x_{12} \\ \sqrt{2}x_{23} \\ \vdots \\ \sqrt{2}x_{13} \\ \vdots \\ \sqrt{2}x_{1n} \end{bmatrix}$$

Moreover, the unique inverse vectorization operation is defined accordingly, such that $X = \mathit{ivec}(\mathbf{x})$.

Based on the above definition, the Frobenius inner product of two matrices is defined below:

Lemma 7.2. *If $X, Y \in \mathbb{S}^{n\times n}$, then*

$$< X, Y >= \mathit{vec}(X)^\top \mathit{vec}(Y)$$

Proof. Let X, Y be defined as (5), then

$$\begin{aligned} < X, Y > =& \mathsf{trace}(X^\top Y) \\ =& x_{11}y_{11} + x_{12}y_{12} + \cdots + x_{1n}y_{1n} + x_{12}y_{12} + x_{22}y_{22} + \cdots + x_{2n}y_{2n} + \cdots \\ & + x_{1n}y_{1n} + x_{2n}y_{2n} + \cdots + x_{nn}y_{nn} \\ =& x_{11}y_{11} + x_{22}y_{22} + \cdots + x_{nn}y_{nn} + 2x_{12}y_{12} + 2x_{23}y_{23} + \cdots + 2x_{2n}y_{2n} + 2x_{1n}y_{1n} \\ =& \mathsf{vec}(X)^\top \mathsf{vec}(Y) \end{aligned}$$

□

References

[1] Z. S. Zhu, A. Su, H. B. Liu, Y. Shang, and Q. F. Yu, "Vision navigation for aircrafts based on 3D reconstruction from real-time image sequences," *Science China Technological Sciences*, vol. 58, no. 7, pp. 1196–1208, 2015, ISSN: 1862281X. DOI: 10.1007/s11431-015-5828-x.

[2] R. Aufrère, J. Gowdy, C. Mertz, C. Thorpe, C. C. Wang, and T. Yata, "Perception for collision avoidance and autonomous driving," *Mechatronics*, vol. 13, no. 10 SPEC. Pp. 1149–1161, 2003, ISSN: 09574158. DOI: 10.1016/S0957-4158(03)00047-3.

[3] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, ser. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2009, ISBN: 9783642039911. [Online]. Available: https://books.google.com/books?id=ixtrCQAAQBAJ.

[4] M. Kampelmühler, M. G. Müller, and C. Feichtenhofer, "Camera-based vehicle velocity estimation from monocular video," *arXiv*, 2018, ISSN: 23318422. arXiv: 1802.07094.

[5] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 4, pp. 271–288, 1998.

[6] K. Yoneda, H. Tehrani, T. Ogawa, N. Hukuyama, and S. Mita, "Lidar scan feature for localization with highly precise 3-D map," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 1345–1350, 2014. DOI: 10.1109/IVS.2014.6856596.

[7] R. Benosman, T. Maniere, and J. Devars, "Multidirectional stereovision sensor, calibration and scenes reconstruction," *Proceedings - International Conference on Pattern Recognition*, vol. 1, pp. 161–165, 1996, ISSN: 10514651. DOI: 10.1109/ICPR.1996.546011.

[8] J. Weng, P. Cohen, and M. Herniou, "Camera Calibration with Distortion Models and Accuracy Evaluation," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 14, no. 10, pp. 965–980, 1992.

[9] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000, ISSN: 01628828. DOI: 10.1109/34.888718.

[10] S. Ganapathy, "Decomposition of transformation matrices for robot vision," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 130–139, 1984, ISSN: 10504729. DOI: 10.1109/ROBOT.1984.1087163.

[11] L. Zou and Y. Li, "A method of stereo vision matching based on OpenCV," *ICALIP 2010 - 2010 International Conference on Audio, Language and Image Processing, Proceedings*, pp. 185–190, 2010. DOI: 10.1109/ICALIP.2010.5684978.

[12] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake, "A robust RGB-D SLAM algorithm," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1714–1719, 2012, ISSN: 21530858. DOI: 10.1109/IROS.2012.6386103.

[13] R. C. Bolles, H. H. Baker, and D. H. Marimont, "Epipolar-plane image analysis: An approach to determining structure from motion," *International Journal of Computer Vision*, vol. 1, no. 1, pp. 7–55, 1987, ISSN: 09205691. DOI: 10.1007/BF00128525.

[14] S. Ullman, "The Interpretation of Structure from Motion," *Proceedings of the Royal Society of London*, vol. 203, no. 1153, pp. 405–426, 1979.

[15] R. Mur-Artal, J. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015, ISSN: 15523098. DOI: 10.1109/TRO.2015.2463671. arXiv: 1502.00956.

[16] F. Mancini, M. Dubbini, M. Gattelli, F. Stecchi, S. Fabbri, and G. Gabbianelli, "Using unmanned aerial vehicles (UAV) for high-resolution reconstruction of topography: The structure from motion approach on coastal environments," *Remote Sensing*, vol. 5, no. 12, pp. 6880–6898, 2013, ISSN: 20724292. DOI: 10.3390/rs5126880.

[17] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 3061–3070, 2015, ISSN: 10636919. DOI: 10.1109/CVPR.2015.7298925.

[18] C. Vogel, S. Roth, and K. Schindler, "View-consistent 3D scene flow estimation over multiple frames," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8692 LNCS, no. PART 4, pp. 263–278, 2014, ISSN: 16113349. DOI: 10.1007/978-3-319-10593-2_18.

[19] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[20] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, 2017. [Online]. Available: http://visual.cs.ucl.ac.uk/pubs/monoDepth/.

[21] Y. Juznietsov, J. Stuckler, and L. Bastian, "Semi-Supervised Deep Learning for Monocular Depth Map Prediction," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, ISSN: 1063-6919. arXiv: 1702.02706. [Online]. Available: https://arxiv.org/abs/1702.02706v3.

[22] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," *Advances in Neural Information Processing Systems*, pp. 1161–1168, 2005, ISSN: 10495258.

[23] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlson, "CNN Features off-the-shelf: an Astounding Baseline for Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2014.

[24] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra R-CNN: Towards balanced learning for object detection," 3, 2019, pp. 821–830.

[25] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe, "Multi-scale continuous CRFs as sequential deep networks for monocular depth estimation," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 161–169, 2017. DOI: 10.1109/CVPR.2017.25.

[26] R. Penrose, "A generalized inverse for matrices," in *Mathematical proceedings of the Cambridge philosophical society*, Cambridge University Press, vol. 51, 1955, pp. 406–413.

[27] J. Cartucho, R. Ventura, and M. Veloso, "Robust object recognition through symbiotic deep learning in mobile robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2336–2341.

[28] M. Tan, R. Pang, and Q. V. Le, *Efficientdet: Scalable and efficient object detection*, 2020. arXiv: 1911.09070 [cs.CV].

[29] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018. arXiv: 1804.02767 [cs.CV].

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2015. arXiv: 1506.02640 [cs.CV].

[31] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. arXiv: 2004.10934 [cs.CV].

[32] J. S. Hunter, "The exponentially weighted moving average," *Journal of quality technology*, vol. 18, no. 4, pp. 203–210, 1986.

[33] C. A. Hall and W. W. Meyer, "Optimal error bounds for cubic spline interpolation," *Journal of Approximation Theory*, vol. 16, no. 2, pp. 105–122, 1976.

[34] J. Friedman, T. Hastie, R. Tibshirani, *et al.*, *The elements of statistical learning*, 10. Springer series in statistics New York, 2001, vol. 1.

[35] L. Ralaivola and F. d'Alché-Buc, "Time series filtering, smoothing and learning using the kernel kalman filter," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, IEEE, vol. 3, 2005, pp. 1449–1454.

[36] F. Chazal, D. Chen, L. Guibas, X. Jiang, and C. Sommer, "Data-driven trajectory smoothing," in *proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2011, pp. 251–260.

[37] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of slam algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, 2009.

[38] J. Sturm, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas, D. Cremers, R. Siegwart, and W. Burgard, "Towards a benchmark for rgb-d slam evaluation," in *Rgb-d workshop on advanced reasoning with depth cameras at robotics: Science and systems conf.(rss)*, 2011.

[39] Z. ChaoQiang, S. QiYu, C. ZHANG, T. Yang, and Q. Feng, "Monocular depth estimation based on deep learning: An overview," *SCIENCE CHINA Technological Sciences*, vol. 63, no. 9, pp. 1612–1627, 2020.

[40] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 7244–7251.