TIPANGLE: A Machine Learning Approach for Accurate Spatial Pan and Tilt Angle Determination of Pan Tilt Traffic Cameras

by

Shreehari Jagadeesha

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2023 by the
Graduate Supervisory Committee:

Aviral Shrivastava, Chair
Aman Arora
Nakul Gopalan

ARIZONA STATE UNIVERSITY

Dececember 2023

ABSTRACT

Pan Tilt Traffic Cameras (PTTC) are a vital component of traffic management systems for monitoring/surveillance. In a real world scenario, if a vehicle is in pursuit of another vehicle or an accident has occurred at an intersection causing traffic stoppages, accurate and venerable data from PTTC is necessary to quickly localize the cars on a map for adept emergency response as more and more traffic systems are getting automated using machine learning concepts. However, the position(orientation) of the PTTC with respect to the environment is often unknown as most of them lack Inertial Measurement Units or Encoders. Current State Of the Art systems 1. Demand high performance compute and use carbon footprint heavy Deep Neural Networks(DNN), 2. Are only applicable to scenarios with appropriate lane markings or only roundabouts, 3. Demand complex mathematical computations to determine focal length and optical center first before determining the pose. A compute light approach "TIPANGLE" is presented in this work. The approach uses the concept of Siamese Neural Networks(SNN) encompassing simple mathematical functions i.e., Euclidian Distance and Contrastive Loss to achieve the objective. The effectiveness of the approach is reckoned with a thorough comparison study with alternative approaches and also by executing the approach on an embedded system i.e., Raspberry Pi 3.

DEDICATION

This work is especially dedicated to my parents Dr. Jagadeesha S N and Sandhya R along with my sister Swathi J Akhilesh. I am only able earn this degree because of their unrelenting support during the darkest hours. From choosing the Problem Statement until the Defense they have nurtured me, motivated me, kept me sane and have shown me path to succeed in completing the thesis and earning the degree. It would be unfair to describe here in words as opposed to the encouragement, love and financial support provided by them. I owe my degree to them. It would also be unfair not to mention my brother-in-law Akhilesh Gargeshwari Surendra and my nephew Anish Akhilesh who put in an effort in making my life easier while pursuing my Masters.

This work is also achieved with a lot of motivation provided by my fellow friends mentioned in no particular order, i.e., Ullas Kalakappa Lagubagi, Sourabh Araga Sunil, Akash Dhananjaya, Abhishek Hanumantha Reddy, Mahesh Balasubramanian, Sai Priyanka Gollu, Anurathi Venkatesh, Harshal Hattaraki, Poojitha Shetty, Kowshik Rangainoor Eswarappa, Sharanya Gulagalale Jayananda, Anagha Dongadi Murali, Pratheek Naidu, Amitabh Das, Rakesh H N, Rohan Nagarad S, Atul Anand, Suchitha Chinnaswamy, Nilesh Aradhya, Sathvik Kirti Kiran, Sujay Giri, Amoghavarsha K P, Karthik Anand Bhat, Srikrishna R Hegde, Sujan Yarasi, Hari Prasad, Naveen M P and Harshith Shankar.

ACKNOWLEDGMENTS

I wish to express my heartfelt gratitude, starting with my committee chair, Dr. Aviral Shrivastava. Not only has he provided me with invaluable feedback on my work, but he has also been a mentor in teaching me how to become a more proficient learner. During moments of discouragement, he motivated me to persevere towards a valuable work, and my two & half years of research under his guidance have instilled in me the profound importance of knowledge and the essence of producing high-quality work.

My deep appreciation goes out to PhD pupil Edward Andert for generously sharing his precious time with me. He was consistently available and offered candid advice concerning my degree, thesis, and career path. I would also like to extend my thanks to the other members of the MPS lab for their amiable companionship and brainstorming sessions.

In addition to chair, I wish to acknowledge faculty committee members Dr. Nakul Gopalan and Dr. Aman Arora for their valuable insights regarding thesis work. Furthermore, I must express my gratitude to my dear friend Matthew Szeto, who has been an unwavering source of support, drawing from his extensive experience within the lab. Also, I am indebted to Kelly O'Brien, my academic advisor, for her guidance on ASU policies, procedures, and requirements.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

Pan Tilt Traffic Cameras (PTTC) are widely used in cities throughout the world. Major metropolitan districts like New York City, Los Angeles, Paris, Berlin, Tokyo, Delhi, Bangalore use significant amount of PTTC to control and modulate traffic and also cater to emergencies. New York metro alone has 868 Traffic cameras installed as per the official website. California has 1170 traffic cameras through the state as per the official website. With so many PTTC distributed across the city, it is difficult for human operators to be monitoring all the PTTCs simultaneously for incidents like accidents, pursuits and traffic stoppages. PAN and TILT angle refers to rotation of camera along horizontal and vertical axis respectively. To solve this problem, many works have been proposed using machine learning concepts and various other image recognition techniques to identify crashes and track traffic flow M *et al.* (2021) Sunny *et al.* (2021). To properly identify the location of a traffic incident, knowing pose of the PTTC plays a vital role along with the GPS co-ordinates. If this information is known then, maps of the city can be created and automatically annotated with incidents. For a stationary mounted camera, this information can be gathered using GPS based survey tools and though it is not fast it does work. However, many traffic cameras are not stationary, i.e., they are pan tilt in nature. These PTTC allow for operators to move the camera pose to an area of interest. The problem is, most of these PTTC do not come with pose encoders to inform the operator what pose they are set to. With a human operator, this is not a downside as the human with knowledge of the city can simply look at the road structure and determine the nearest crossroads to figure out the pose. For an algorithm that is detecting a traffic incident, this is prohibitive as the human skills needs to be involved in the process to identify where the traffic event occurred and annotate in the map or send and update to responders. This is therefore restrictive in making a useful traffic monitoring system, and an inexpensive, accurate and quick method is needed to map pose of these moving PTTC.

In today's rapidly evolving world, the ever-growing number of vehicles on the road has raised profound concerns about traffic management and safety. Traffic Control Divisions worldwide are increasingly turning to digitalization to address these challenges, leveraging technology to revolutionize the way we handle traffic-related issues. From reducing congestion and enhancing safety to improving incident response times, digitalization has become an indispensable tool for modern traffic management.

One of the key advantages of digitalization is its capacity to ease congestion on our roads. PTTC, an integral part of this transformation, are now ubiquitous, with New York, alone boasting a staggering 868 of these advanced cameras as per the official website. These cameras help authorities monitor traffic flow, detect accidents, and efficiently notify emergency services, ensuring that critical interventions occur swiftly. Monitoring every pan tilt traffic camera live is, however, not practically possible for traffic monitoring personnel due to the sheer volume of cameras in operation.

To address this challenge, an effective solution is localizing live traffic data onto a map. This innovative approach can offer a comprehensive overview of traffic conditions in real time, aiding traffic management personnel in making data-driven decisions. Moreover, localization empowers traffic management divisions to track anomalies in traffic flow, enabling data-driven decisions that enhance traffic light management and rerouting strategies. This flexibility proves invaluable, especially during mass gatherings like match days, where it can help ease congestion and improve overall traffic management. Real-time traffic data also enables the estimation of travel times without relying on actual GPS data from vehicles. This not only enhances navigation for the general public but also aids law enforcement in tracking suspect vehicles within a city without engaging in high-speed chases. With the limited availability of police personnel for pursuit, the ability to track multiple vehicles simultaneously is an essential tool for maintaining public safety. Furthermore, the rise of autonomous vehicles adds a new dimension to traffic management. By tracking these vehicles, Motor Vehicle Departments (MVDs) can ensure that autonomous systems are functioning properly and have not gone rogue, contributing to safer roads. Additionally, the integration of digital tools enable traffic management divisions to effectively document incidents and rapidly notify emergency services. This streamlined communication and response system has the potential to save lives and reduce the impact of traffic-related accidents.

1.1 Problem Statement:

In the context of traffic surveillance, the pose of traffic cameras can undergo various changes for several reasons. One primary factor contributing to these pose variations is the deliberate adjustment made by traffic monitoring personnel, who may temporarily alter the camera's orientation to track construction activities at a specific site. However, such changes are not always restored to their original positions, leading to potential discrepancies in monitoring accuracy. Additionally, natural elements such as wind and rain can inadvertently shift the camera's position, further complicating data integrity. To address these challenges and maintain the accuracy of vehicle localization on the map, accurate pose estimation of the PTTC is necessary. Notably, these cameras often lack built-in sensors to autonomously determine their pose relative to the world. Hence this thesis delves in solving the problem of estimation of pose of PTTC.

The thesis presents to solve the problem using SNN. Detailed investigation have been carried out on the numerical properties, convergence characterics and it's ability to accurately match the most similar image and subsequently determine the Pan and Tilt angle from its metadata.

1.2 Organization of Thesis:

The embodided work in this thesis has been arranged in 6 chapters. In Chapter 2, Review on some of the state-of-the-art techniques employed so far for calculating the Pan and Tilt angles of PTTC is conducted. These include determination of focal length and optical centre of a PTTC, usage of lane marking, usage of elliptical formula for roundabout and also use of Deep Neural Network(DNN) approaches for determining pose of the PTTC. In the next chapter i.e., Chapter 3 a brief introduction to SNN is carried out along with a thorough discussion on the mathematical concepts involved in determining pose of the PTTC using SNN. In chapter 4 architecture of the SNN devised and procedure in calculating the pose of PTTC is explained in detail. In chapter 5 experimental results of the devised aprroach along with comparison study against latest DNN is examined. In chapter 6 the enhancements done so as to improve the execution speed of approach on an embedded system i.e., Raspberry Pi 3 is explained.

Chapter 2

RELATED WORKS

A brief survey of the earlier work done in the field of determining the pose of PTTC has been reviewed and included in this chapter.

2.1 Literature Survey:

Recently, there has been a surge in high-performing supervised camera data segmentation methods. These methods typically combine spatial features and semantic information usually, embedded in a Regular Convolutional Neural Network. The segmentation masks are inherently steered towards more probable foreground objects, such as cars, cyclists or pedestrians, using networks trained on large databases such as ImageNet (Deng *et al.*, 2009) as a backbone. This allows them to achieve very accurate segmentation results in nearly any scenario, given that at least one manually annotated frame is available (Tsai *et al.*, 2016),(Caelles *et al.*, 2017).

(Lu *et al.*, 2019) propose an online random forest for mapping and sports player detection to deal with moving foreground objects. Modern unsupervised segmentation methods are also able to accurately detect such objects by making additional assumptions about object motion or the camera setup (Wehrwein and Szeliski, 2017), but still they typically require the entire camera data to be processed before the output is generated.

The segmentation methods described above are suitable for applications such as automatic video fine-grained sports analysis (Allebosch *et al.*, 2019). However, for domains such as automatic surveillance or obstacle detection in traffic, there is a need for immediate, real-time detection of anything that can be considered noteworthy and for these purposes it is essential that the system takes accurate input data for training and validation so that these systems do not fail and cause further escalations.

Although Deep Learning i.e., using GoogLeNet, ResNet and EfficientNet can be used in solving the mentioned problem statement, they come with fair share of challenges. The main drawbacks of using Deep Neural Network Architecture are as follows,

- Extensive labeled data is a prerequisite for deep learning. For instance, the development of autonomous vehicles necessitates the availability of millions of images and thousands of hours of video footage(Zohuri and Moghaddam, 2020).
- Substantial computational resources are a vital component of deep learning. High-performance GPUs, designed with a parallel architecture optimized for deep learning tasks, can significantly accelerate the training process when used in conjunction with clusters or cloud computing(Zohuri and Moghaddam, 2020).

(Pollefeys *et al.*, 1996) developed a technique to calibrate the variable focal length (the distance between the camera lens and image sensor) of a pinhole camera model batch across photos. The focal length and optical centre (the center point of the image) position parameters were improved at each image location for a batch optimization over several images using a method based on bundle adjustment (Heyden and Astrom, 1997). Adding to this, (Pollefeys *et al.*, 1998) described a technique to identify useful trajectories for self-calibration. In both works, self-calibration in a batch setting is examined.

A plane-based method for calibrating pinhole camera models in a batch scenario was introduced in (Sturm and Maybank, 1999). For each camera, the focal lengths and optical centre are estimated. (Lourakis and Deriche, 2000) described a technique for self-calibrating pinhole cameras utilizing restrictions derived from the Singular Value Decomposition of the basic matrix between two views. The focal length and optical centre parameters of each camera are then obtained by solving these constraints in a nonlinear batch setting. Given that the rotation of the camera is known (Frahm and Koch, 2003), proposed a method to calibrate the variable intrinsics (mainly focal length and optical centre) of a pinhole camera in a batch scenario.

(Song and Tai, 2006) propose automatic calibration of PTTC. The authors use image processing for the lane marking on the road to find the Focal length of the camera. Subsequently, the pan and tilt angles of the PTTC are obtained by using the estimated focal length. This proposed method eliminates the need for manual intervention in determining the locations of distinctive features. Instead, it autonomously utilizes a combination of parallel lane markings and lane width to calculate essential camera parameters, including focal length, tilt angle, and pan angle. Image processing techniques have been devised to automatically detect and identify parallel lane markings. However, the main drawback of this estimation is necessary requirement of lane markings needed for the estimation of focal length of the PTTC. Thereby making this approach irrelevant at intersections/roads which do not have accurate lane markings.

(Nagayoshi and Pollefeys, 2014) proposed a method for pose estimation which needs a marker set on a camera and tracks it to obtain circular trajectories generated by pantilt motions of the camera. The marker is tracked and several-hundred marker points are generated from one pan-tilt motion. As a result, the camera pose can be estimated. The circular trajectories in 3D space are observed as ellipse in images. A normal vector and the center of the circle are estimated from the ellipses. A refinement process to minimize differences between modeled pan-tilt motions and observed three or more circular trajectories is then performed to achieve higher accuracy. However the main drawback is that this approach requires the use of multiple cameras facing the target to track the trajectory of the target. Thereby, needing the use of high resolution Full HD tracking cameras to observe the marker placed on the target.

(Dinh and Tang, 2014) presented an approach for calibrating cameras in roundabout traffic scenes. The method suggests utilizing a circular landmark to achieve calibration. By analyzing the image, the proposed method derives the ellipse equation and compares it with the projected equation of the corresponding circle in real-world traffic scenes. The calibration outcomes can subsequently be utilized for various purposes, such as estimating vehicle speed, facilitating vehicle tracking, and deriving vehicle trajectories for roundabout traffic data collection. The main problem with this approach is that it is only applicable for roundabout scenarios and not regular intersections.

As it can be seen from various studied approaches, determination of pose of the PTTC is dependent on estimation of intrinsic properties (focal length and optical centre) which is a tedious and expensive process or require multiple cameras or is applicable in certain scenarios only like roundabouts. As opposed to the previous works, we here present a novel, simple, dependable and scalable approach TIPANGLE, to determine the pose of the PTTC based on Euclidian Distance and Contrastive Loss mathematical concepts implemented in the SNN. This approach has been used successfully used in weakly supervised metric learning (Bromley *et al.*, 1993) and face verification (Chopra *et al.*, 2005). The cited paper (Bromley *et al.*, 1993) outlines an algorithm designed to authenticate signatures made on a pen-input tablet. This algorithm relies on a SNN, comprising two identical sub-networks connected at their outputs. Throughout the training process, these sub-networks extract features from two distinct signatures. Simultaneously, a connecting neuron computes the distance between the two feature vectors. The verification process involves comparing an extracted feature vector with a stored feature vector associated with the signer. Signatures that fall within a specified threshold of similarity to the stored representation are considered genuine, while all others are categorically rejected as forgeries. The general idea is to learn a function that maps input patterns into a target space such that the norm in the target space approximates the "semantic" distance in the input space.

The above approach is also utilized for a facial verification objective (Chopra *et al.*, 2005), where the training procedure aims to reduce a discriminative loss function. This loss function encourages the similarity metric to be minimized for facial pairs from the same individual and maximized for pairs from different individuals. The transformation from the initial data space to the desired space is accomplished through a convolutional neural network specifically crafted to handle geometric distortions with robustness in mind.

Chapter 3

METHODOLOGY

3.1 Dataset Collection:

A dataset containing 195 images were meticulously collected using the Canon 1300D camera, equipped with an inclinometer mounted on top to precisely measure the tilt angle. The pan angle measurements were obtained using the tripod's built-in angle measurement, ensuring the accuracy of the dataset. The setup is shown in Fig.1. These images were collected at a vantage point on the bridge over the University Dr to cover the University Dr and Rural Rd intersection. These images were collected systematically in 3 sets i.e., day, dusk and night scenario with each set consisting of 65 images. The Pan angle varies from 0° to 30° with a step size of 2.5°. The Tilt angle varies from -0.5° to -10.5° with the same step size of 2.5°. Table 1 shows the grid containing pan and tilt angles of the images collected. Fig.2 shows three images collected at the same pan and tilt angle during day,dusk and night. In addition, 90 separate images are collected during random times, have been reserved for testing.

0.0°,-0.5°	2.5°,-0.5°	...	15.0°,-0.5°	...	27.5°,-0.5°	30.0°,-0.5°
0.0°,-3.0°	2.5°,-3.0°	...	15.0°,-3.0°	...	27.5°,-3.0°	30.0°,-3.0°
0.0°,-5.5°	2.5°,-5.5°	...	15.0°,-5.5°	...	27.5°,-5.5°	30.0°,-5.5°
0.0°,-8.0°	2.5°,-8.0°	...	15.0°,-8.0°	...	27.5°,-8.0°	30.0°,-8.0°
0.0°,-10.5°	2.5°,-10.5°	...	15.0°,-10.5°	...	27.5°,-10.5°	30.0°,-10.5°

Table 1. Pan and Tilt Angles of the Collected Images

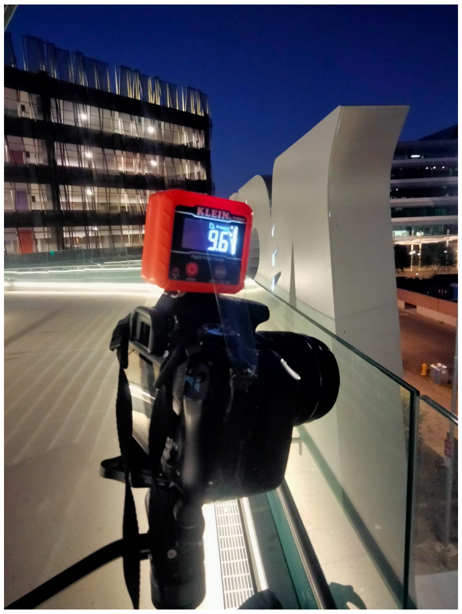


Figure 1. Setup Used to Capture the Images

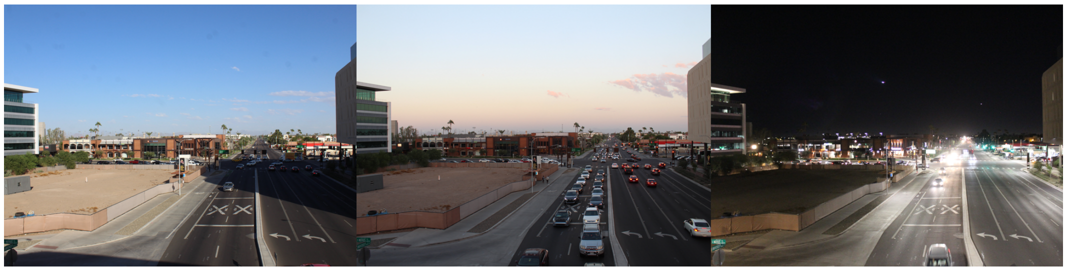


Figure 2. Images Taken at Same Angle I.E., 0° Pan and -3° Tilt Angle During Day, Dusk and Night Scenarios

3.2 Siamese Neural Network and Training:

Similarity learning is a supervised machine learning technique in which the goal is to train the model to learn a similarity function, which measures how similar two objects are and returns a similarity value. When the images or objects are similar, a high score is returned, and a low score is returned when they are not. SNN scale well with small amounts of data. A SNN extracts features from the input training data into the feature space using a loss function preferably triplet loss or contrastive loss.

Then euclidean distance is used to measure the nearest matching feature in the feature space. The best similarity match occurs when euclidean distance is zero or is close to zero when compared to counter inputs Bromley *et al.* (1993). Catering to our problem statement, a quick response from the system is needed so as to notify emergency services. Collecting a large number of images from the PTTC units proves challenging, as their poses are frequently changed by the Motor Vehicle Division. Therefore, we have chosen the Siamese network as the optimal approach to address our problem. This paper presents our proposed approach, providing a detailed explanation of the methodology, the relevant mathematical concepts involved, and experimental results.

The computational problems associated with the DNNs, namely, GoogLeNet, ResNet and EfficientNet can be mitigated by using the SNN. In this, the estimation is based on the calculation of Euclidian Distance and Contrastive Loss during training and only Euclidian Distance during inference. These functions are computationally light and works with skewed data as well. Little attention has been paid in the literature so far, towards the application of the SNN for solving the camera pose estimation problem. Inference of this approach can be easily implemented and executed on an embedded system i.e., Raspberry Pi 3 in our use case. In this chapter, the mathematical concepts involved in solving our problem stetement using SNN has studied and explained in detail.

Convolutional Neural Networks (CNN) are multilayer neural networks that are made up of several convolutional layers with varying kernel sizes that are interlaced by pooling layers that summarize and downsample the output of its convolutions before feeding it to subsequent layers which include activation functions and pooling.

On the other hand, SNN (Cartwright (2015)) is a type of network architecture that consists of two identical subnetworks. The concept of SNN draws inspiration from the phenomenon of Siamese twins, a term originating from the conjoined twins Chang and Eng Bunker, born in Siam (now Thailand) in the early 19th century. Just as these twins were conjoined yet distinct individuals, Siamese networks are designed to compare data points and determine their similarity or dissimilarity, using the power of neural networks. The twin subnetworks which are usualy CNN or Artificial Neural Networks (ANN) or Recurrent Neural Networks (RNN) are configured in the same way, with the same parameters and shared weights. Parameter updates are mirrored across both subnetworks. Siamese networks seek to maximize the similarity between the output feature vectors of the two branches. This framework has been used successfully in weakly supervised metric learning (Chopra *et al.*, 2005) and face verification (Schroff *et al.*, 2015). A loss function (Cartwright (2015)) at the top connects these subnetworks by computing a similarity metric (Cartwright (2015)) based on the Euclidean distance between the feature representations on each side of the Siamese network. A differentiable loss function is typically chosen so that gradient descent and network weight optimization can take place. Given a differentiable loss function, back propagation is used to update the weights of different layers. Either Triplet Loss Schroff *et al.* (2015) or Contrastive Loss Kumar *et al.* (2023) function is chosen in the case of SNN.

Euclidean Distance which is obtained using pythagorous theorem between two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ in a two dimension space is given by,

$$\text{Euclidian Distance}(D_w) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.1}$$

Similarly Euclidian Distance between two points $P = (x_1, x_2, \ldots, x_n)$ and $Q = (y_1, y_2, \ldots, y_n)$ in an n-dimensional space is given by,

$$\text{Euclidian Distance}(D_w) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.2}$$

We have chosen Contrastive Loss function in our application of SNN. The reason for not using Triplet Loss and choosing Contrastive Loss instead is explained in further down.

A distance metric δ for Contrastive Loss function must have the following properties,

- Non-negativity: $\delta(x, y) \geq 0$
- Identity of Non-discernibles: $\delta(x, y) = 0$ implies $x = y$
- Symmetry: $\delta(x, y) = \delta(y, x)$
- Triangle inequality: $\delta(x, z) \leq \delta(x, y) + \delta(y, z)$

The Contrastive Loss Chopra *et al.* (2005) is defined as follows:

$$L\left(s_1, s_2, y\right) = \alpha(1-y)D_w^2 + \beta y \max\left(0, m - D_w\right)^2 \tag{3.3}$$

where s_1 and s_2 are two samples (in this instance, street images), y is the similarity label which either zero or one indicating whether or not the two samples belong to the same class. y is zero if both the images belong to the same class and 1 if both the images belong to different class. α and β are two constants, and m is the margin. $D_w = \left\|f\left(s_1; w_1\right) - f\left(s_2; w_2\right)\right\|_2$ is the Euclidean distance computed in the embedded feature space, f is an embedding function that maps a street image to real vector space through CNN, and w_1, w_2 are the learnt weights for a particular layer of the underlying network. The SNN aims to bring the output feature vectors closer for similar input pairs and distance the feature vectors away for dissimilar input pairs by assigning binary similarity labels to pairs (Kumar *et al.* (2023)). Each branch of the Siamese network can be thought of as a function that embeds the input image in space.

Triplet Loss is not chosen on purpose as it would require an anchor image for each PAN and TILT angle i.e., an anchor image is required in creation of every triplet (positive image, negative image and anchor image) and would only increase the complexity of the approach.

Hence, the Contrastive Loss (3.3) function has been chosen such that, images of nearby PAN and TILT angles will be closer to each other than images with largely spaced PAN and TILT angles.

3.3 Inference:

The network has two input fields for comparing the two patterns and one output with a state value representing the similarity between the two patterns. Using the Euclidean distance as the measure during the inference, smaller value of Euclidiean Distance between two image embeddings that are closely similar or identical would result in a viable solution. Instead of iterating all 195 images and finding their distance between input image and then sorting for the least distance, Gradient Descent approach has been used here for faster inference.

The algorithm is shown in Fig.3. The algorithm takes as input an initial seed point which is assumed to be the center image in the grid, the input image, and a grid containing pan and tilt angles. The grid in Table is shown in the It operates through an iterative process, continuously exploring neighboring points and updating its position based on the smallest distance found. The key steps in this algorithm include initializing the local minimum with the original seed, calculating distances to

```
Algorithm 1: Gradient Descent Algorithm
Input  : OriginalSeed, InputImage, The Grid with x,y coordinates
Output: LocalMinima
LocalMinima ← OriginalSeed;
CurrentSeed ← OriginalSeed;
SmallestDistance ← distance to CurrentSeed from input image;
distances[];
while True do
    Using CurrentSeed, append distances[] of 8 neighboring images
     individualy if each is unexplored;
    Sort(distances[]);
    if distances[0] < SmallestDistance then
        LocalMinima ← x,y location of distances[0];
        SmallestDistance ← distances[0];
    if distances[0] < 2*SmallestDistance then
        CurrentSeed ← x*,y* location of distances[0];
    distances[].pop();
    if length(distances[]) == 0 then
        return LocalMinima;
```

Figure 3. Gradient Descent for Inference

neighboring points, sorting these distances, and updating the local minimum if a closer point is found. The algorithm also adapts the current seed to move closer to the local minimum if the distance is sufficiently small. It maintains a record of explored points. The process continues until a termination condition i.e., when the data structure preferably an array containing the distances to the neighbouring images is empty or all the distances of the neighbouring images are twice the local minima. This algorithm represents a simplified form of gradient descent, serving as a foundation for finding local minima within images.

0.0°,-0.5°	...	12.5°,-0.5°	15.0°,-0.5°	17.5°,-0.5°	...	30.0°,-0.5°
0.0°,-3.0°	...	12.5°,-3.0°(0.76)	15.0°,-3.0°(0.63)	17.5°,-3.0°(0.87)	...	30.0°,-3.0°
0.0°,-5.5°	...	12.5°,-5.5°(0.84)	15.0°,-5.5°(0.29)	17.5°,-5.5°(0.74)	...	30.0°,-5.5°
0.0°,-8.0°	...	12.5°,-8.0°(0.88)	15.0°,-8.0°(0.52)	17.5°,-8.0°(0.98)	...	30.0°,-8.0°
0.0°,-10.5°	...	12.5°,-10.5°(0.97)	15.0°,-10.5°(0.92)	17.5°,-10.5°(0.95)	...	30.0°,-10.5°

Table 2. Gradient Descent Search for Reduced Inference Time

Chapter 4

EXPERIMENTATION AND RESULTS

Fig.4 shows the SNN used by us. Each image is flattened into a 1D vector by concatenating pixel values to derive a vector of pertinent features for calculation of euclidiean distance. The length of these vectors corresponds to the total number of dimensions, which is equivalent to the number of RGB pixels within the image. For an image recognition problem, we use a CNN architecture inspired by Krizhevsky *et al.* (2012). We list the filter sizes for convolution and pooling layers as H*W*D, where H is the height, W is the width and D is the depth of the corresponding filter. Stride denotes the distance between the application of filters for convolution and pooling operation. Rectified Linear Units (ReLU) are used as the activation function to the output of all convolutional and fully connected layers throughout the network. Local Response Normalization is used to generalize the learnt features, as described in Krizhevsky *et al.* (2012), with the parameters shown in the corresponding rows in Table 3 and Table 4.

In the pursuit of developing an effective deep neural network architecture for the mentioned specific task of determining the pose, the approach began with the creation of a more streamlined model. This involved systematically exploring various network configurations to strike a balance between performance and computational efficiency.

The journey to develop an effective SNN architecture shown and specified in Table 3, Table 4 and Fig.4 for determining pose commenced with the construction of a minimalist model. The initial focus was on creating a small-scale architecture,

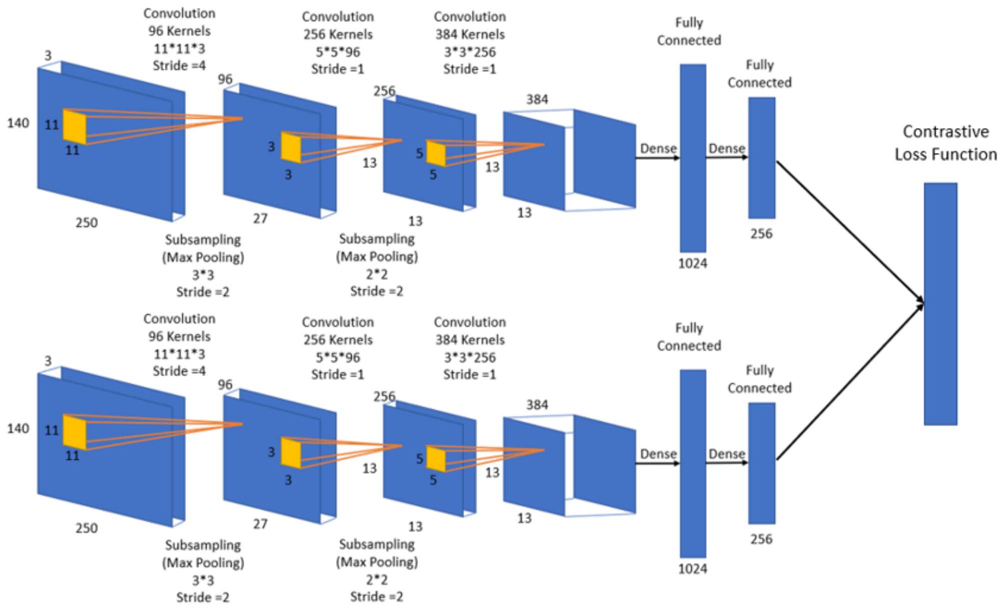


Figure 4. Siamese Neural Network

Table 3. Network Architecture

Layers	**Kernel Size**	**Stride**	**Padding**	**Input Channels**	**Output Channels**	**Activation Function**
Convolutional Layer 1	11x11	4	0	1	96	ReLU
Max Pooling Layer 1	3x3	2	-	-	-	-
Convolutional Layer 2	5x5	1	0	96	256	ReLU
Max Pooling Layer 2	2x2	2	-	-	-	-
Convolutional Layer 3	3x3	1	0	256	384	ReLU
Linear Layer 1	-	-	-	384	1024	ReLU
Linear Layer 2	-	-	-	1024	256	ReLU

Table 4. Parameters

Parameters	**Values**
Activation Function	ReLU
Learning Rate	0.0005
Optimizer	Adam Optimizer
Loss Function	Contrastive Loss with
Loss Margin	2.0
Batch Size	64
Epoch Count	100

prioritizing simplicity and ease of computation while maintaining a optimal level of performance.

Beginning with this foundational approach, the network was intentionally designed with just two Convolutional layers and one Max Pooling Layer. Although the model exhibited computational efficiency, it became evident that its capacity to capture nuances within the data was restricted. Consequently, the performance metrics fell short of the desired performance level.

To address these limitations, the architecture was then expanded to the configuration presented in Table 3, Table 4 and Fig.4. This enhanced version integrated additional max-pooling and convolutional layer, increasing its depth and potential to discern more complex features. This resulted in promising results, with the network exhibiting improved performance.

However, in the quest for further improvements, a more ambitious approach was explored by augmenting the architecture even more. An additional max-pooling layer and convolutional layer were introduced, aiming to enhance the network's ability to extract nuances in the data. Despite meeting the problem requirement, this configuration came at a cost. The increased complexity resulted in greater computational overhead .The model size affects the speed of inference as well as the computing source it would consume. Ultimately, this drawback outweighs the marginal performance enhancements achieved as the aim was to run the inference model on a embedded system. While the initial minimalist approach demonstrated computational efficiency, it compromised on performance. The architecture configuration shown and presented in Table 3, Table 4 and Fig.4 struck the right balance between model complexity and performance gains. The larger configuration, while achieving performance, proved to be computationally expensive.

In the pursuit of constructing an effective SNN architecture for the intricate task of estimating the pose of traffic cameras, every parameter was chosen to contribute to the network's learning and prediction prowess. The network's architectural components were systematically fine-tuned to attain a balance between feature extraction capabilities and dimensionality reduction.

The choice of kernel size is a critical factor in performance of the model. Smaller kernels like (1, 1) or (3, 3) are good at capturing fine image details, while larger ones like (11, 11) are better for capturing larger, more abstract features. Selecting the right kernel size involves empirical experimentation (Nielsen (2018)).

Beginning with the Convolutional Layer 1, which features an 11x11 kernel size and a stride of 4, the architecture captures large, foundational features such as road and building structures. The subsequent integration of Max Pooling Layer 1, with a 3x3 pooling size and a stride of 2, increases the network's ability in extracting such large features. Convolutional Layer 2, with a 5x5 kernel size and stride of 1, further refines feature extraction, particularly those of intermediate sizes, like traffic lights and sidewalks. Max Pooling Layer 2, operating with a 2x2 pooling size and a stride of 2, is added further for dimensionality reduction.

Using an odd-sized filter in Convolutional Layers ensures that all the preceding layer pixels are symmetrically positioned around the output pixel. This symmetry is crucial because without it, we would need to address distortions that can occur when using an even-sized kernel. Hence, to maintain implementation simplicity, even-sized kernel filters are often avoided. To visualize convolution as an interpolation process from the available pixels to a central pixel in words, it becomes apparent that an even-sized filter does not allow for straightforward interpolation to a central pixel(Nielsen (2018))

This hierarchy of Convolutional and Max Pooling layers collectively culminates in an architecture capable of effectively discerning both substantial and intricate features inherent in traffic camera images, thereby optimizing its efficacy for absolute pose estimation. The usage of Rectified Linear Unit (ReLU) activation function is chosen to introduce non-linearity while mitigating the vanishing gradient issue. By setting negative values to zero, ReLU facilitates efficient gradient flow during back propagation.

Beyond the architecture, the chosen parameters continue to wield well over the network's learning trajectory and predictive potency. The learning rate is set at 0.0005 for cautious and yet efficient convergence. This calibrated learning pace, empowers the model to adapt its weights without leading into erratic divergent behavior. The Adam optimizer recognized for its adaptive learning rate mechanism is chosen since Adam optimizer dynamically customizes learning rates for individual parameters. This dynamicity inherently expedites convergence, particularly important while navigating gradients that exhibit variance across the network.

The simplified contrastive loss function Hadsell *et al.* (2006) is defined as,

$$(1-Y)\frac{1}{2}\left(D_W\right)^2+(Y)\frac{1}{2}\left\{\max\left(0,m-D_W\right)\right\}^2 \tag{4.1}$$

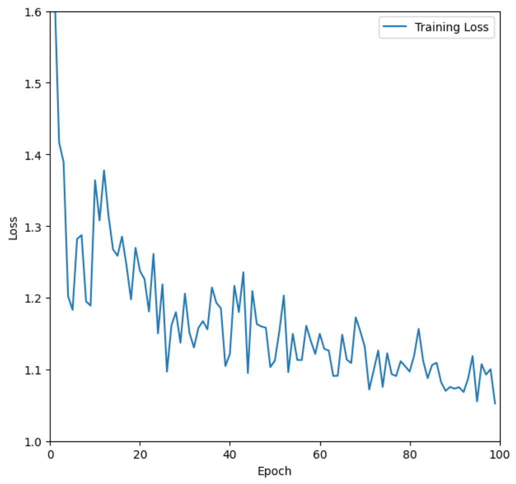


Figure 5. Training Loss History

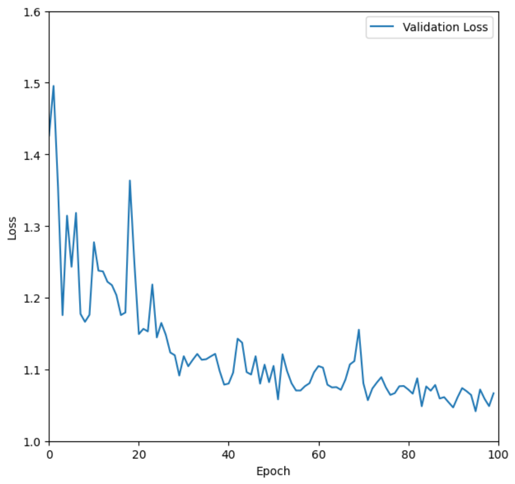


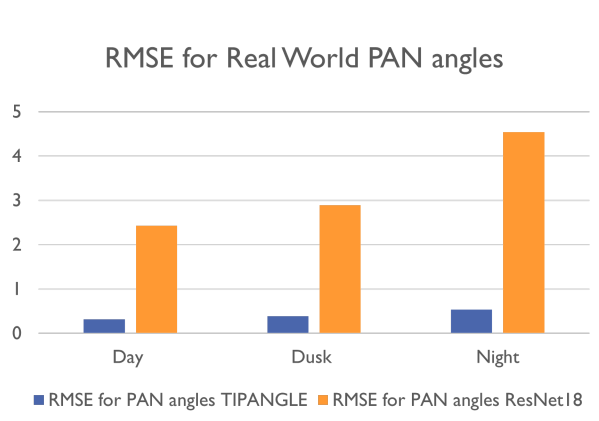
Figure 6. Validation Loss History

where Dw is the Euclidean distance between the sister networks' outputs. Y is the similarity label. If the first and second images are from the same class, the value of Y is zero; otherwise, the value of Y is one. The images are said to be of same class if they have same pan and tilt angles. m is a margin value greater than zero. The presence of a margin indicates that dissimilar pairs that are further apart will not contribute to the loss. We have set our margin to two through empirical approach for better results.

Furthermore, the operational parameters i.e., the batch size set at 64 and epoch count set at 100 facilitates computational efficiency and exhaustive learning. The batch size harnesses the parallel processing capabilities of hardware, balancing between acceleration and stability during training.

In the previous chapters mainly mathematics and architecture behind SNN were discussed in detail, this chapter reckons the use of SNN for the problem statement instead of DNN with exprimental data.

We trained the model for 100 epochs, using training rate equal to 0.005, and



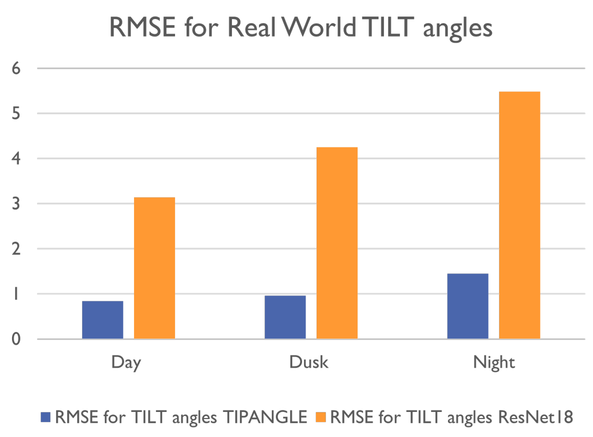


Figure 7. Pan and Tilt Angle RMSE Results for Tipangle and Resnet18

minimum batch size equal to 64. Our entire framework is implemented using Tensorflow and Keras libraries. The training was carried out using a Intel HD graphics and a Core i5 Cpu which took 6 minutes and 48 seconds to train the network. Summarizing, the model demonstrates a positive learning trend, with decreasing Contrastive Loss over the course of training and validation. The training is stopped when The Contrastive Loss almost stagnates and does not decrease any further. The results suggest that the model effectively learns from the training data and generalizes well without being overfit, as evidenced by the improvement in both training and validation Contrastive Loss metrics shown in Fig.5 and Fig.6.

In order to demonstrate the ability of the proposed system to accurately detect the pose of the PTTC, the Root Mean Squared Error (RMSE) values between predicted and actual values of output angles are shown juxtaposed with ResNet18 to comment on real world accuracy and provide insight on TIPANGLE prowess.

It can seen from the Fig.7 that the RMSE is minimum during the day. As the lighting decreases the RMSE value increases, however the proposed solution still performs good with minimal errors. When the results are compared with ResNet18 it can clearly seen that TIPANGLE shows better results i.e., it performs better than DNN in accuracy.

Table 5. Comparison of Network Architectures

Network Architecture	No of Layers	Training Time in s	Execution Time in s
TIPANGLE	7	408	0.829
ResNet-18	18	2246	3.713

The Table 5 compares two neural network architectures, TIPANGLE and ResNet-18, both run on a Colab CPU. TIPANGLE has 7 layers, requiring 408 seconds for training and 0.829 seconds for execution. In contrast, ResNet-18 with 18 layers takes 2246 seconds to train and 3.713 seconds for execution. These metrics offer insights into the computational efficiency of each architecture, aiding in selecting the appropriate model based on training and execution time constraints.

Using Pinhole Camera estimation in order localize the vehicles Pose estimation of the camera is necessary. Locaization of vehicles using TIPANGLE results are shown in Fig.8 further as well to demonstrate the prowess of TIPANGLE.

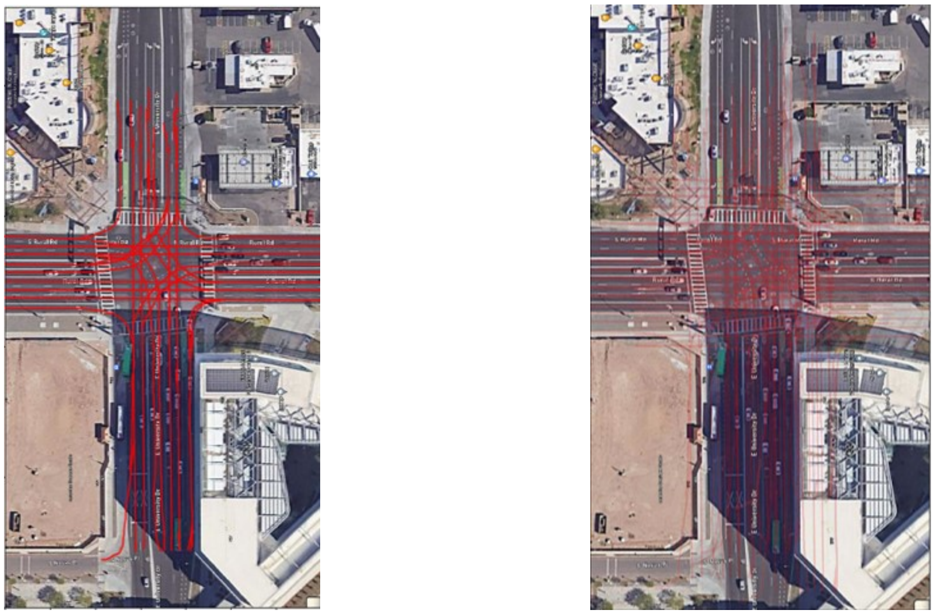


Figure 8. Localization with and without TIPANGLE

Chapter 5

SPEEDING UP THE SYSTEM FOR EMBEDDED SYSTEM IMPLEMENTATION

Convolutional Neural Network models can be resource-intensive, requiring large amounts of memory and computational power to train and use for inference. This can make it difficult to deploy these models on resource-constrained devices such as Raspberry Pi shown in Fig. 9 with limited computational power(Quad Core 1.2GHz Broadcom BCM2837 64bit CPU) and RAM(1.0 Gb). To overcome this limitation, we use separable convolution, pruning and parallel execution to reduce these model's resource requirements.

Separable convolution involves two steps: first, a depth-wise convolution groups input features by channel, treating each channel individually and producing outputs with the same number of channels. Second, a point-wise convolution uses 1×1 kernels to capture relationships between channels efficiently, reducing computations by half as opposed regular convolution.

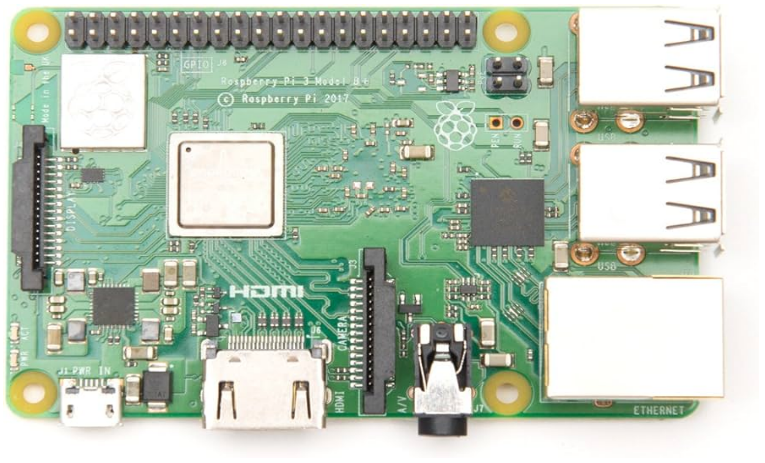


Figure 9. Raspberry Pi 3

Pruning reduces neural network complexity and size by removing specific weights. In our approach, PolynomialDecay pruning schedule was used with these parameters: initial sparsity of 0.50 (half of weights pruned at the start), final sparsity of 0.90 (90% of weights pruned by the end), and pruning initiated from training start (begin step=0). Pruning occurred every 100 steps during training epochs, gradually increasing model sparsity over time, balancing compression and performance preservation.

Since euclidean distance has to be calculated between input image and one image of all the poses to find the accurate pose, parallel execution would be greatly beneficial. Raspberry pi 3 has a quad core processor and to exploit the complete prowess of the processor, inference execution is multi-threaded to 4 threads. This further drastically reduced the execution time to 40s from earlier 60s. Further increasing the number of threads to 8 and 16 did not yeild in performance but increased the overhead instead and increased the execution time to 45s and 48s for 8 and 16 threads respectively.

Chapter 6

CONCLUSION

Most traffic PTTC do not include sensors such as encoder or Inertial Measurement Unit so as to know the direction in which they are facing. An approach is proposed to predict the absolute PAN and TILT angle of PTTC so that further processing can be done especially by modern Machine Learning Systems. It is very important to know the vital information regarding the traffic PTTC such as its GPS location and direction of the camera to make use of the data appropriately generated by traffic PTTC. If a vehicle is in pursuit or an accident has occurred at an intersection using valid data from camera is necessary to quickly figure out where the incident has occurred. As more of these functions are automated with AI it becomes critical to have this information generated automatically otherwise the data output of the AI system is not useful.

Hence, A system is proposed here to detect the direction in which the traffic PTTC are facing using a minimal amount of training data so that minimal human time is required to collect the data. In the process of proposing the approach multiple stateof-the-art existing approaches were studied in depth with their shortcomings. The proposed approach reckons its prowess by overcoming those shortcomings explained in chapter 2. The concept of SNN was studied in depth. In implementation of the approach an empirical method was chosen after multiple iterations of finding the balance between performance and computational expense. A new dataset was collected and labeled upon on which the proposed appraoch was trained. A thorough comparitive study was conducted with the latest DNN architectures. The results present the prowess against latest DNNs in solving the same problem. To even further prove the pwoess of the approach localization was tried with TIPANGLE and without TIPANGLE. The approach's prowess is further proven by executing it on an embedded system. An already efficient approach is made even faster using pruning, parallel execution and using computationally efficient neural network implementations.

REFERENCES

Allebosch, G., D. Van Hamme, P. Veelaert and W. Philips, “Robust pan/tilt compensation for foreground–background segmentation”, Sensors **19**, 12, URL https://www.mdpi.com/1424-8220/19/12/2668 (2019).

Bromley, J., I. Guyon, Y. LeCun, E. Säckinger and R. Shah, “Signature verification using a "siamese" time delay neural network”, in “Proceedings of the 6th International Conference on Neural Information Processing Systems”, NIPS’93, p. 737–744 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993).

Caelles, S., K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers and L. Van Gool, “One-shot video object segmentation”, in “2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)”, pp. 5320–5329 (2017).

Cartwright, H., *Artificial Neural Networks*, vol. 1260 (2015).

Chopra, S., R. Hadsell and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification”, in “2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)”, vol. 1, pp. 539–546 vol. 1 (2005).

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database”, in “2009 IEEE Conference on Computer Vision and Pattern Recognition”, pp. 248–255 (2009).

Dinh, H. and H. Tang, “Simple method for camera calibration of roundabout traffic scenes using a single circle”, Intelligent Transport Systems, IET **8**, 175–182 (2014).

Frahm and Koch, “Camera calibration with known rotation”, in “Proceedings Ninth IEEE International Conference on Computer Vision”, pp. 1418–1425 vol.2 (2003).

Hadsell, R., S. Chopra and Y. LeCun, “Dimensionality reduction by learning an invariant mapping”, in “2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)”, vol. 2, pp. 1735–1742 (2006).

Heyden, A. and K. Astrom, “Euclidean reconstruction from image sequences with varying and unknown focal length and principal point”, in “Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition”, pp. 438–443 (1997).

Krizhevsky, A., I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in “Advances in Neural Information Processing Systems”, edited by F. Pereira, C. Burges, L. Bottou and K. Weinberger, vol. 25

(Curran Associates, Inc., 2012), URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Kumar, A. J., A. Manvitha, B. Poojitha, M. Mahitha, K. Ashesh and K. V. D. Kiran, “Optimising face recognition system using contrastive learning and contrastive loss”, in “2023 7th International Conference on Computing Methodologies and Communication (ICCMC)”, pp. 251–256 (2023).

Lourakis, M. I. A. and R. Deriche, “Camera self-calibration using the kruppa equations and the svd of the fundamental matrix: The case of varying intrinsic parameters”, (2000).

Lu, J., J. Chen and J. Little, “Pan-tilt-zoom slam for sports videos”, in “British Machine Vision Conference”, (2019).

M, B. K., A. Basit, K. MB, G. R and K. SM, “Road accident detection using machine learning”, in “2021 International Conference on System, Computation, Automation and Networking (ICSCAN)”, pp. 1–5 (2021).

Nagayoshi, H. and M. Pollefeys, “Estimating camera pose using trajectories generated by pan-tilt motion”, in “2014 2nd International Conference on 3D Vision”, vol. 1, pp. 561–568 (2014).

Nielsen, M. A., “Neural networks and deep learning”, (2018).

Pollefeys, M., R. Koch and L. Van Gool, “Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters”, in “Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)”, pp. 90–95 (1998).

Pollefeys, M., L. Van Gool and M. Proesmans, “Euclidean 3d reconstruction from image sequences with variable focal lengths”, in “Computer Vision — ECCV ’96”, edited by B. Buxton and R. Cipolla, pp. 31–42 (Springer Berlin Heidelberg, Berlin, Heidelberg, 1996).

Schroff, F., D. Kalenichenko and J. Philbin, “Facenet: A unified embedding for face recognition and clustering”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)”, (2015).

Song, K.-T. and J.-C. Tai, “Dynamic calibration of pan–tilt–zoom cameras for traffic monitoring”, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **36**, 5, 1091–1103 (2006).

Sturm, P. and S. Maybank, “On plane-based camera calibration: A general algorithm, singularities, applications”, in “Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)”, vol. 1, pp. 432–437 Vol. 1 (1999).

Sunny, S. M., T. Rahman, S. M. Zohurul Islam, A. Mujtaba, K. F. Ahmed and S. Saha, “Image based automatic traffic surveillance system through number-plate identification and accident detection”, in “2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)”, pp. 467–472 (2021).

Tsai, Y.-H., M.-H. Yang and M. J. Black, “Video segmentation via object flow”, in “2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)”, pp. 3899–3908 (2016).

Wehrwein, S. and R. Szeliski, “Video segmentation with background motion models”, in “Proceedings of BMVC”, (2017).

Zohuri, B. and M. Moghaddam, “Deep learning limitations and flaws”, Modern Approaches on Material Science **2** (2020).