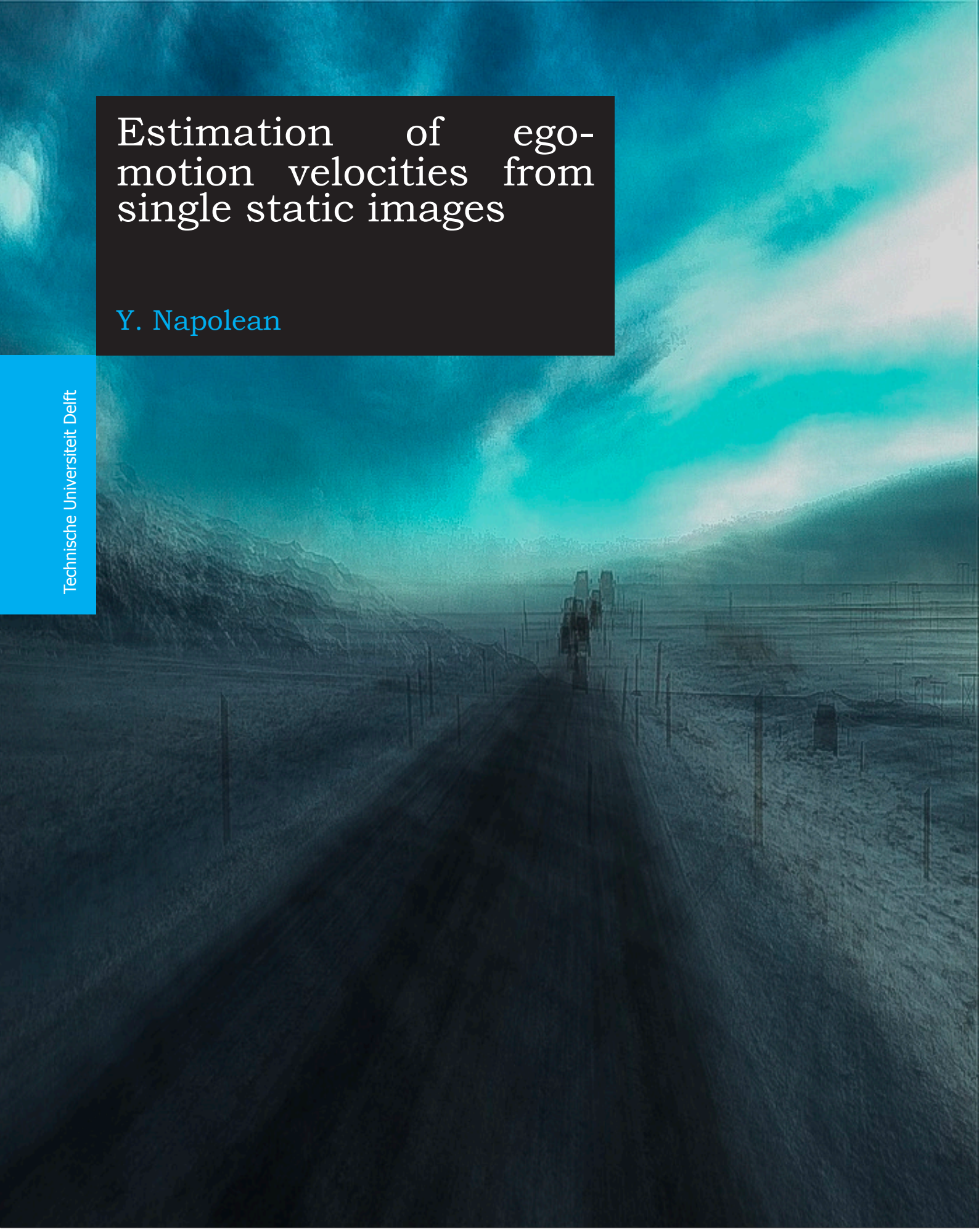

Estimation of ego-motion velocities from single static images

Y. Napolean

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

Estimation of ego-motion velocities from single static images

by

Y. Napolean

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology

Student number: 4521277
Project duration: June 2017 to May 2018
Thesis committee: Dr. G. C. H. E. de Croon, Supervisor & Chair
Dr. Ir. J. V. C. van Gemert, Supervisor
Dr. Ir. E. van Kampen, Committee member

TU Delft Delft University of Technology

Preface

My time in TU Delft has been nothing short of an amazing journey, culminating with a topic I find truly fascinating and I write this document in the hope that it captures my intrigue and transfers it to the reader as well. This work would not have been possible without Dr. Ir. Jan van Gemert and Dr. G.C.H.E. de Croon who were kind, considerate and supportive. They provided me with invaluable guidance while still affording me the freedom to explore the topic on my own. I am truly grateful to have them as my supervisors. I would also like to express my heartfelt gratitude to Dr. Ir. Erik-Jan van Kampen for his time and consideration.

To adapt to a new environment is not an easy task. I was lucky to be accompanied by great friends on my journey towards an MSc, great friends like Naina who is wise enough to counsel me in distress, yet juvenile enough to plan pranks with. Being a Happy-go-lucky sort of person. I frequently tend to get complacent with myself, which is when Shubham steps in with his skepticism to ensure I am always on my toes. I also owe a lot to Siva Sankar and Satya who have stuck with me through thick and thin for almost a decade now.

I was fortunate to meet wonderful people in Delft as well. Kindered Spirits like Greeshma and Deepak, who have given me a fresh perspective of life. I thank Vishal for listening to my rants and for the occasional life hacks he gave. I would also like to thank Datta, Balaji Sridharan, Manjunath, folks from the C&S department (shout out to Simon Vrouwenvelder, Master Kwadwo and Peter Luteijn) and the PRB Group,EWI for their support and encouragement.

Living with people you are not familiar with, seems a weird proposition to most. I have had the privilege of staying with amazing people like Florian who has helped me shrug off many a worry, Livia who I look up to but might never emulate and Andrea, with whom I have had many interesting machine learning related discussions. All of them have been the epitome of benevolence, and have helped me find my home away from home.

Last but in no way the least, I would like to thank my parents. They have always been the pinnacle of understanding and have supported me throughout this endeavour and my whole life. I will always be indebted to them.

Y. Napolean
Delft, May 2018

The cover image was generated by the author using the Python programming language and the Python Imaging Library(PIL), with pictures captured while travelling on the Ring Road in Iceland. All the captured images were averaged to simulate the motion blur effect and edited in Adobe PhotoShop LightRoom.

Contents

Scientific Article

Estimation of Ego-motion Velocities from Single Static Images

Yeshwanth Napolean
Student
Delft University of Technology

J.C. van Gemert
Supervisor
Delft University of Technology

G.C.H.E de Croon
Supervisor
Delft University of Technology

Abstract

Velocity estimation based on visual information is a well-researched topic. Traditional approaches usually rely on how a given feature or features change between successive images in a sequence. However, a single static image might contain motion information that could potentially be leveraged to estimate the optical flow. It can be hypothesized that motion blur and context of the scene are two sources of motion information in static images. This research work has two main goals, one is to investigate the prospect of using a learning-based framework to model a mapping directly to camera ego-motion velocity. The second goal is to analyze the contributing features in learning such a mapping. Experiments show that the model is able to learn velocity based on context of the scene but performs better when input images contain motion blur.

1. Introduction

Given the steady progress in the field of robotics and automation and its widespread acceptance and use in day to day life, it has become increasingly vital to estimate velocity for exploration, navigation and related tasks. With the advent of self-driving cars, autonomous unmanned aerial vehicles and other similar systems, it is now of paramount importance to obtain accurate velocity estimates of the agent to prevent accidents. There are many methods to measure the velocity of a given agent, such as GPS (Global Positioning System) or using inertial measurement units. However, errors accumulate in IMUs [1], and thus they need a supplementary system such as GPS for correction. Also, GPS as a standalone system has poor performance in some indoor scenarios [19]. However, cameras are cost-effective, compact and low power while still providing robust and reliable data [23]. Vision based sensors have been used in a wide range of applications including velocity estimation,

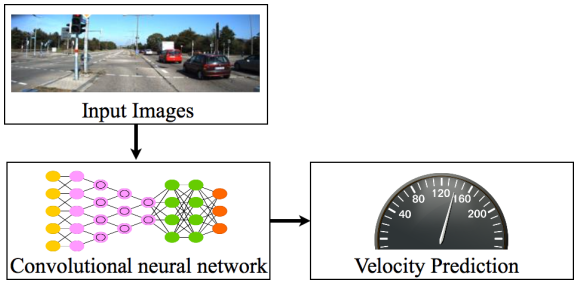

Figure 1. Velocity estimation model. The deep learning model is trained on the input images to predict velocity.

especially on board unmanned aerial vehicles. These applications include obstacle avoidance, velocity estimation [26], integration with an inertial navigation system [35] and navigation [42]. Another one of the main advantages of cameras as sensors apart from being information rich is that they can function independently of the platform/agent that they are deployed on. For instance, state estimation (using potentiometers) in humanoid agents is quite different from that employed on-board unmanned aerial vehicles(using an IMU). However, both bipedal and aerial robots can utilize vision based state estimation.Thus, in recent times drawing inspiration from examples in nature, several computer vision based approached have been put forth for velocity estimation [2] [13][16].

The handling of visual data in an effective manner was made easier with the introduction of convolutional neural networks (CNNs) [21] [22]. Following the seminal work of Alex Kriezehvsky et. al in 2012 [20] on the ImageNet object recognition challenge [6], CNNs gained popularity. Using GPU acceleration made deep neural networks computationally more tractable. Convolutional neural networks have since been used to learn how to predict optical flow [7] with significant accuracy. Convolutional neural networks have also been used for heterogeneous blur removal [12] and learning to classify blur kernels [37]. Research has also

been carried out to estimate velocity from blur present in images [24].Research in computer vision has already previously investigated motion estimation from single static images [29] via structured regression with a random forest. Thus, it is possible that motion blur contains velocity information and blur kernel identification can be achieved with a learning based framework. The architecture for the learning based model for velocity estimation is depicted in Figure 1. Research work on image based velocity estimation has increased in recent years. The predominant method for velocity estimation from images is by using optical flow which was first introduced by an American psychologist named James Gibson . Optic flow is the apparent motion of texture in the visual field relative to the camera and can be utilized to obtain velocity and distance information [15]. Optic flow and motion blur are both related to motion. The salient difference is that motion blur occurs in a single frame while optical flow is the motion between frames.

The key contribution of this body of work is twofold - i) The investigation of using single static images as input to a CNN feature extractor and then a feed forward network for the task of velocity estimation. ii) Analyzing the features learned by the convolutional neural network and ultimately quantifying the influence of motion blur and spatial context as a feature in learning a mapping from pixel space to velocity space.

The rest of the article is organized as follows, section 2 presents literature survey of relevant research for motion blur estimation and vision based velocity estimation. However, there is very little work done regarding spatial context as a feature for velocity estimation, thus to provide additional perspective, the role of context in relevant tasks are presented. Section 3 deals with the methodology undertaken, discussing the network architecture and approach to analyzing the deep learning model. Section 4 presents the results obtained and the final section presents the conclusions and recommendations for future research.

2. Related Work

Motion blur is usually characterized by a point spread function. This point spread function is defined by the motion angle and length. The recovery of the point spread function is a particularly challenging task owing to the short duration of the image degradation process and loss of information it causes. However, study of motion blur PSF has revealed that it exhibits a unique behavior in the frequency domain [30]. Blur identification with spectral nulls making use of power spectrum and the power cepstrum was investigated [34], but this approach fails when there is a large amount of noise or if the blur size is small. Auto-correlation based methods for PSF estimation have also been investigated [27] [44]. While the aforementioned methods achieve reasonable performance, the key issue with them is that they all require incorporation of domain knowledge in the framework, engineered features or pre-processing. Thus, we make use of deep learning models which eliminates the need for feature engineering.

Recent research in deep learning has shown that, the motion blur parameters can be estimated from raw images itself rather than resorting to frequency domain analysis. CNNs have particularly been used for blur identification [37]. The motion space is discretized and the convolutional neural network is trained as a classification model to estimate probabilities of motion kernels for each patch of the image, then dense motion blur kernels for the whole image is estimated using a Markov random field. Fully convolutional neural networks have also been used to estimate motion flow from motion blur in an image [12]. This approach does not require any post processing. Training and testing of the model is directly performed on the whole image, exploiting additional spatial context to estimate a dense motion flow map accurately. However, here for motion flow estimation, the fully convolutional network is trained over discrete outputs. Discretization of the motion space could pose an issue(loss of information) as motion is an inherently continuous quantity. While these research endeavours have achieved significant results in blur estimation, they focus on the removal of blur. We aim to understand the role of motion blur in the task of velocity estimation.

Traditional methods for optical flow estimation take a variational approach [17][41] [31]. The disadvantage to these methods are that they require engineered methods for matching, aggregation and interpolation. Research has also been carried out on the topic of estimation of optic flow using machine learning algorithms. Sun et. al [5] analyze the underlying statistics of optical flow, thereby proposing a steerable random field to model the statistical relationship between image and flow boundaries.Taylor et. al [39] proposed a convolutional restricted Boltzmann machine capable of extracting low-level motion features which is ultimately used for action recognition. The common issue with these approaches are that they work with controlled experimental setup and do not have performance comparable to classical flow estimation algorithms on real-world data [7].

The neural network architecture termed 'FlowNet' put forth by Fischer et. al [7] is capable of learning to predict optical flow with good generalization capability. The architecture proposes a CNN trained end to end to compress information spatially and then refined to obtain optical flow prediction. Ilg et. al [18] further built upon 'FlowNet', proposing a new framework 'FlowNet 2.0' which includes a stacked architecture with the ability to estimate optical flow for small and large displacements. While FlowNet architectures utilize image pairs, Walker *et al.* showed that it is possible to estimate dense optical flow from a single image frame [40]. The article hypothesizes that the learned optical flow rep-

resentation is context dependent but does not quantify the relationship. FlowNet 2.0 has good performance on optical flow estimation, but it utilizes successive image frames. We aim to build on the work by Walker *et al*. quantifying the role of scene context.

As human beings, we unwittingly use contextual information on a daily basis. We can situate ourselves in an environment and process information based on things around us. Studies in Cognitive Psychology [3] provide evidence that contextual cues such as relative size and location play a significant role for object detection in humans. This is relevant when dealing with velocity estimation because we would tend to predict higher velocities while travelling on highways, as compared to a road winding through a mountain pass. High-level contextual information has been shown to augment low-level features for object detection tasks [11] achieving better performance. Context has also been shown to play a vital role in 3D scene understanding [46]. However, when it comes to velocity estimation, motion blur and ultimately motion flow have been investigated as a cue/feature for velocity estimation, but the role of context is not very well explored.

3. Methodology

As human beings, the primary visual cues we generally use to estimate velocity are motion blur and context. When encountering obstacles in forward motion, we predict that our velocity would be lower to manoeuvre around the obstacle. When objects appear more blurred, we tend to assume that we are travelling faster. Given that convolutional neural networks are based on the mammalian visual cortex, it is reasonable to hypothesize that context and motion blur play a part in velocity estimation with artificial neural networks. When a camera is subject to motion within an exposure period, the illumination changes are integrated over time and the sharpness is smeared over the image, thus forming motion blur. In other words a captured image can be thought of as an averaged sample over a time period, as a result, moving objects in that time period cause motion blur. Image deblurring to recover the original image is an actively researched topic [12][37] since traditional computer vision tasks such as segmentation and tracking are difficult to perform without knowing the blur kernel. However, the motion blur being removed actually has motion information [32]. Thus, it is possible that a model could learn a function that maps motion blur parameters to velocity of the camera

An image with motion blur retains information that parameterizes the blur. This enables the recovery of motion from a single static image. Motion blur in an image can be characterized by its Point Spread Function (PSF). A motion blurred image (b) can be thought of as a convolution (*) of the unblurred image (i) and the point spread function (h) with additional noise (μ).

$$b(x, y) = i(x, y) * h(x, y) + \mu(x, y) \tag{1}$$

The point spread function is defined as:

$$h(x, y) = \begin{cases} \frac{1}{L}, & \text{if } \sqrt{x^2 + y^2} \leq \frac{L}{2} \text{ and } \frac{x}{y} = -tan(\phi). \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Where ' L ' is the motion length and ϕ is the motion direction. The length of the motion blur is proportional to the relative velocity between the object and the camera.

A review of literature has shown single image frames do contain temporal information which could possibly be utilized to estimate the ego-motion velocity. One key assumption made is that the features in the scene have on average, the same distance. The first step would in this research work would be to train a convolutional neural network to predict the velocity given single static image frames.

This section presents and motivates the relevant technical details of the methodology undertaken. The first subsection discusses the choice of network architecture. The second section presents the concept of transfer learning which will be used to evaluate the feature learned by the deep learning model.

3.1. Network Architecture

An appropriate convolutional neural network architecture must be determined for the task of velocity estimation. Given the dynamic nature of the computer vision and deep learning research fields, several convolutional architectures have been developed. These models such as ZFNet [45], GoogleNet [38], VGG [36] and ResNet [14] have all achieved a significant level of performance in the ILSVRC Image classification challenge. Among the above, the VGG16 network architecture has been prevalent, deployed for several tasks such as image style transfer [9], remote sensing image classification [4] and has been showed to learn a generalized set of features which could be attributed to the depth of the network and the fact that it has pooling layer once after every two convolutional layers. The data is relatively less downsampled, allowing it to store more information. The VGG16 architecture is presented in Fig. 2

The VGG architecture is slightly modified for this scenario. The convolutional layers and the first two fully connected layers are kept intact, the final fully connected layer is modified to output one number - the velocity prediction, rather than the 1000 class scores for ImageNet, the last loss layer is changed from Softmax loss to a Euclidean Loss function which is more suited for the regression task.

$$L = \frac{1}{2N} \sum_{i=1}^{N} ||\hat{y}_n - y_n||_2^2 \tag{3}$$

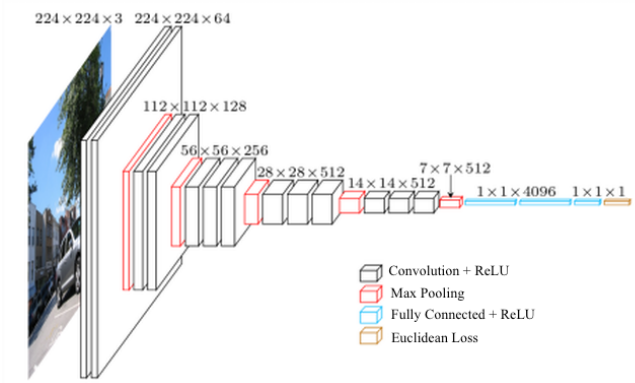

Figure 2. VGG16 Convolutional neural network Architecture [28]

The network hyperparameters are tuned until an optimal combination is reached. The KITTI data is fed as input into the VGG16 CNN. The model is trained for 100,000 iterations. The trained model weights are frozen and the model is tested with the test data.

3.2. Transfer Learning

The training of deep models can be unwieldy especially when there is a relatively less data. This has lead to the popularity of the transfer learning paradigm. The previously learned features serve as a starting point, which could enable faster convergence to the solution. Initial layers in the convolutional network architecture learn to detect corners and edges and the subsequent layers hierarchically learn finer features. Considering that the source domain for the pre-trained VGG16 model is an object recognition problem, the feature space of the domain and the predictor function learned are completely different when compared to the task of velocity estimation. However, since the initial layers possess the ability to detect lower level features, it would prove useful.
Transfer learning in convolutional neural networks can be implemented in two ways :

- **Fine-tuning the model**: The update of network weights of the pre-trained model resumes through backpropagation on the new task.It is possible to fine-tune the whole network, or keep the initial layers frozen, tuning subsequent layers to prevent overfitting.
- **Freezing the weights**: The model is used as a fixed feature extractor, no weight updates occur in this scenario.

Each of the above approaches is useful in a specific situation. Freezing the weights can be used to keep all factors equal and evaluate the model. Fine-tuning can help in reduction of computational cost. In the context of this research work, The model is fine-tuned on the KITTI and vKITTI datasets from the pre-trained ImageNet VGG16 weights. The learned model is then used as a feature extractor on other tasks to obtain a better understanding as to what features the network learned.

4. Experiments & Results

To efficiently train deep learning models, a large amount of labelled data is required. More specifically, images captured by a moving camera, annotated with the instantaneous velocity at the time of image capture would constitute the dataset for training this deep learning model. Due to the increasing popularity of autonomous driving there are many such datasets available such as the comma.ai dataset [33], the KITTI dataset [10] and the Oxford RobotCar dataset[25]. Among these datasets, the KITTI dataset presents itself as the ideal candidate in this scenario as it has a synthetic, generated counterpart name vKITTI [8] which proves useful to evaluate the role of motion blur. There is no mention of a blur model applied for the creation of the vKITTI dataset, the role of context can be evaluated using the vKITII dataset.
The vKITTI dataset is a clone of five videos selected from the original KITTI dataset. The creators of the dataset also automatically generate modified versions of the original sequence(different weather and imaging conditions). The vKITTI dataset does not directly have information of ground truth velocity for the generated scenes, but the extrinsic parameters which capture the transformation from 3D world coordinates to camera coordinates are available. The extrinsic matrix includes the rotation matrix(R) and the world coordinate system origin represented in camera coordinates. Thus, the camera pose can be obtained by the equation,

$$P = -R^T T \tag{4}$$

Where P is the camera postion, R^T is the transpose of the camera rotation matrix and T is the translation vector, the latter two are obtained from the camera extrinsics matrix. The temporal difference of successive camera positions will result in the ground truth velocities.
Since vKITTI is a synthetic dataset generated without a motion blur model. However, for validation, a frame is extracted from KITTI along with its corresponding vKITTI frame. The gradient magnitude of the two images are calculated and depicted in Fig. 3. It can be seen that the edges are more clear in vKITTI than with KITTI. Also the median of the top 0.1% gradient value is higher for vKITTI than KITTI as shown in Table 1.
To evaluate the role of context in the estimation of camera motion, a dataset with images from a variety of places is required. Scene recognition datasets would be best suited for this task. Two of the biggest datasets for scene recognition are the SUN database [43] and the MIT places database

Video Frame	Gradient
KITTI	791.64
vKITTI	892.02

Table 1. Comparison of amount of blur in image. The median value of the max 0.1% of gradients is higher in vKITTI. The lower value for KITTI implies it contains motion blur.

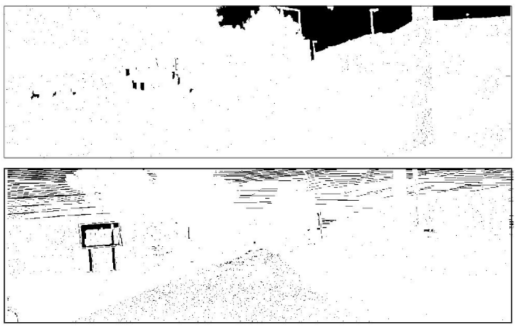

Figure 3. Gradient magnitude of KITTI (top) and corresponding vKITTI frame(bottom). Black corresponds to lower gradient values, white implies higher gradient. The clear edges visible in the gradient of vkitti, show that it has no motion blur.

[47]. These large datasets share a few common classes but the MIT places dataset is larger, diverse and well documented so it is taken under consideration.

The aforementioned datasets aid in establishing the role of motion blur and context in velocity estimation. However, these datasets are recorded with cars and thus the motion is planar - cars are incapable of rolling motion along its own axis. Thus it would be interesting to investigate velocity estimation with images recorded on board an unmanned aerial vehicle. Such a dataset(drone collected images annotated with velocity) is not available to the best of the authors knowledge, so it must be created. The paparazzi open source autopilot and the motion capture system at Delft University of Technology (Opti-track) are utilized along with the Parrot Bebop drone for the creation of this dataset.

4.1. Exp 1: Evaluating performance of deep model

To investigate if a deep learning model can estimate velocity with good preformance and to analyze what the model learns, it is necessary to first train the VGG network to predict velocities.

Before quantifying the role of motion blur and context, training the network on a different task, and utilizing the features extracted from this network for velocity estimation would present itself as a good baseline, offering additional perspective to the performance of the model trained on other tasks. To this end, the VGG16 network is trained to detect pedestrians and cyclists in images via bounding box regression, The networks weights are then frozen and the KITTI

Figure 4. Example Images from the KITTI dataset.

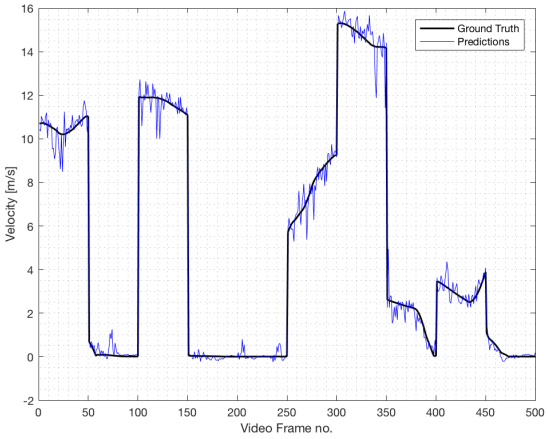

Figure 5. Predictions and corresponding ground truth on the KITTI test set.

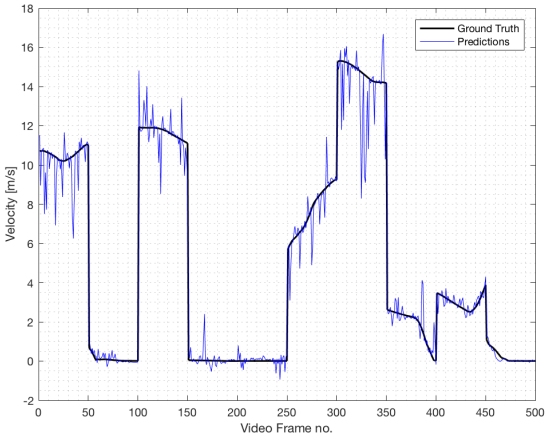

Figure 6. Prediction and ground truth - model trained on different task. Performance worse than KITTI trained model.

image features from the penultimate fully connected layer are extracted. The extracted features are then trained with

a feed forward neural network for the task of velocity estimation. The results are presented in Figure 6. It is apparent from the results that the predictions have poorer performance in this case than the pervious scenario when features were extracted from the model trained on the KITTI dataset.

4.2. Exp 2: Evaluating motion blur as a feature

To estimate the contribution of motion blur in establishing a mapping from pixel space, to motion space, the trained VGG16 network is used as a feature extractor for a motion blur regression task. The task is to estimate the motion blur angle and length from the images. The input to the model are images from the vKITTI dataset convolved with motion blur kernels.The predictions on the test set are shown in Figure 8.

Figure 7. An Example frame from the vKITTI dataset convolved with the blur kernels. From the top, the first frame is the original, the second is convolved with a kernel of parameters (len,theta) = (30,10), the third frame is convolved with the kernel of parameters (45,20) and the final one is convolved with the kernel of parameters (55,30)

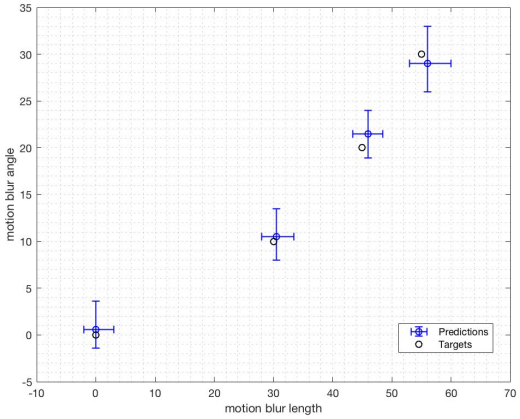

Figure 8. Per kernel mean and variance of prediction along with ground truth.

It can be seen that the model is able to predict motion blur with reasonable accuracy, thus it is evident that the model learns features that capture motion blur information.

4.3. Exp 3: Evaluating scene context as a feature

For analyzing the role of context, a sample of 1.8 million images from the MIT places dataset are supplied as input to the trained convolutional neural network. If the network has learned scene based information, it should predict higher values for highway like scenes in MIT places. However, this does not seem to be the case as can be seen in Figure 9 The predictions from the network have high per class standard deviation. The magenta lines represent predictions on images from indoor classes such as hotel room while the black lines represent outdoor classes like highway road.

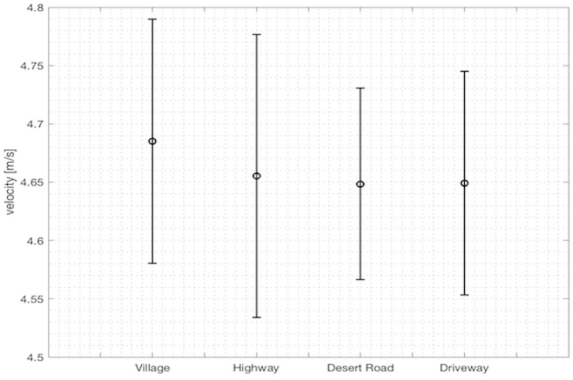

Figure 10. Mean and variance of predictions on relevant scenes

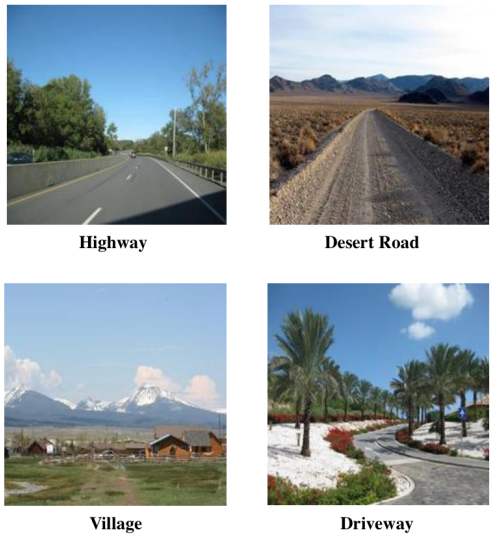

Figure 11. Example Frames from relevant classes of the MIT places dataset.

However, it might be worth noting that some classes have

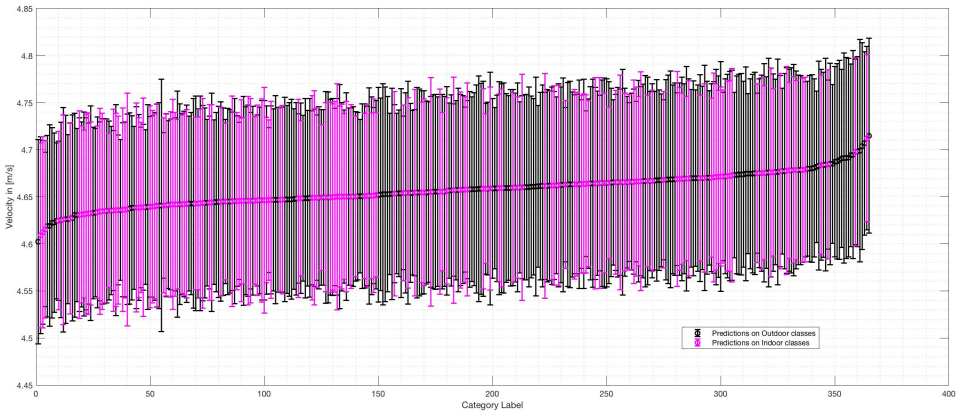

Figure 9. Velocity predictions on the MIT places dataset

pictures taken indoors and outdoors and there are completely new sets of features that the network has previously not been exposed to.

To offer some perspective, the predictions on a few pertinent classes have been depicted in Figure 10.

It can be seen that village has a higher speed prediction as compared to highway. While the indoor classes seem more densely distributed towards the lower predictions, there is no discernible pattern in the mean values of predictions among different classes.

4.4. Exp 4: Validation of role of motion blur and context

While initial evidence points to the conclusion that velocity estimation is more dependent on motion blur than it is on context, it is necessary to design additional experiments to completely validate this hypothesis. Cross-testing of KITTI and vKITTI trained models would offer additional information to the nature of the mapping learned for velocity estimation. To preserve scene context and carry out an objective evaluation, only the frames utilized for cloning from the KITTI dataset are used.

Table 2. Testing across KITTI and vKITTI

Experiment	Mean Absolute Error [m/s]
Trained and tested on KITTI	3.15
Trained and tested on vKITTI	4.83
Trained on KITTI, tested on vKITTI	6.70
Trained on vKITTI, tested on KITTI	6.81

It is evident that training and testing on the real world KITTI dataset has the least mean absolute error. Training and testing on vKITTI has a slightly higher error, possibly because there is no motion blur information in the features learned by the vKITTI network and the network learns the predictor function based on context information only. However, training on KITTI and testing on vKITTI and vice versa, the performance drops further. This can be attributed to the fact that while training with KITTI, the model learns motion blur related features which are then not available on the test set. The model is unable to adapt to this, leading to larger errors. When training with the vKITTI dataset, the network learns spatial context related features but motion blur information is present in the test set. The network is unable to utilize the additional information. The motion blur could potentially be acting as noise, rendering the predictions worse.

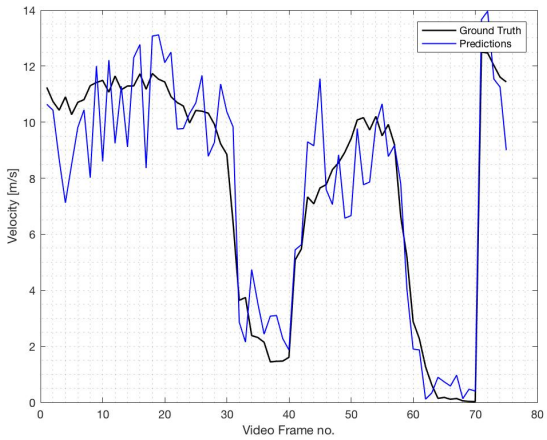

Figure 12. Predictions and Targets - training on blurred vKITTI

It would be of significance to study the effect of introducing motion blur in the vKITTI dataset, to see how it affects the performance of the model. Blur is introduced in the vKITTI dataset by averaging two successive frames and repeating the same temporal averaging on the velocity to obtain the new ground truth. The predictions obtained for training and testing the model on the blurred vKITTI dataset are shown in Figure 12.

Training on the blurred vKITTI dataset is observed to have slightly worse performance (MSE = 5.3 m/s) to that of training without the blur. This could be due to the fact that the motion blur in real images is heterogeneous in nature and the motion information is not completely captured by averaging consecutive images.

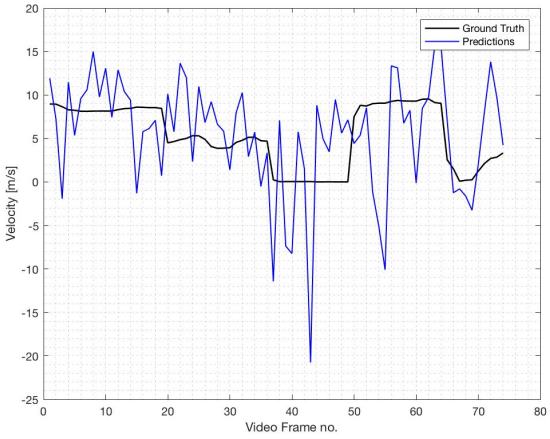

Figure 13. Predictions and Targets - training on KITTI, testing on Robotcar

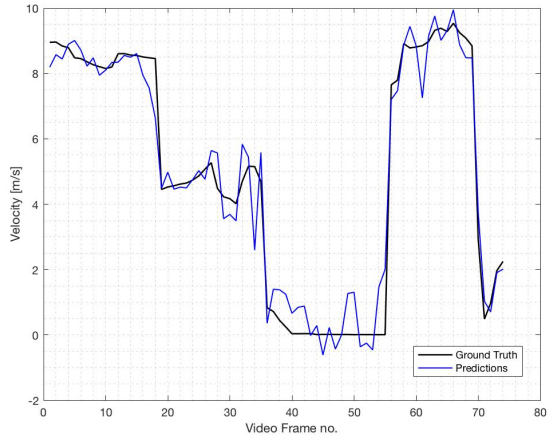

Figure 14. Predictions and Targets - training and testing on Robotcar. Improvement in performance as compared to KITTI trained model.

It is also necessary to understand the capability of the model to determine its limitations. To that end, the model is trained on the KITTI dataset and tested on the oxford RobotCar dataset [25]. The oxford dataset is not only recorded in different scenes but also has higher gradation in illumination change. While a 'vanilla' convolutional neural network is usually not capable of domain adaptation, it would still be relevant to investigate if the model is able to learn a generalized representation of motion. The model is trained and tested on the oxford dataset as well.

The results are presented in Figure 13 and Figure 14. The model does not show any domain adaptation capability but the model has good performance on the oxford dataset. It is only slightly worse than the performance on the KITTI dataset - with an MAE of 4.06 which could be because of the larger range of illumination change.

4.5. Exp 4: Model performance on drone data

The datasets tested so far involve planar motion, the cars and thus the cameras all move on a road, there is no upward or downward motion. To test how the model handles another axis of motion, images are captured and the velocity is logged on board a Parrot bebop drone. Not only is there an additional axis of motion, which could result in higher order of blur heterogeneity, there is probably a subtler contextual dependency since the dataset is recorded in a controlled environment. For datasets recorded in the world, there is scene information directly relating to velocities such as traffic lights, road structure and so on. This dataset is fed to the model and the results are presented in Figure 16.

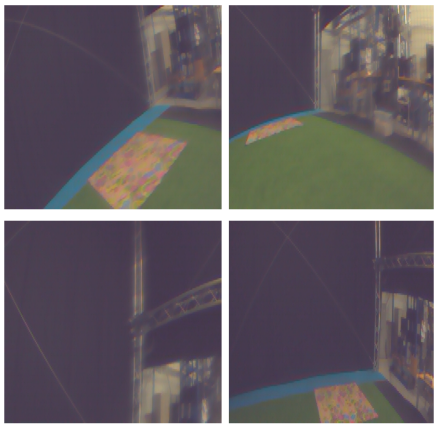

Figure 15. Example Images from the drone dataset

The mean absolute error for the drone dataset is found to be 0.51, while the error is lower here, an objective evaluation would also consider the maximum speed attained by the drone which is around $2m/s$. Comparing the prediction error and the maximum speeds in all the datasets(max speed for KITTI is $13m/s$), it can be seen that while the er-

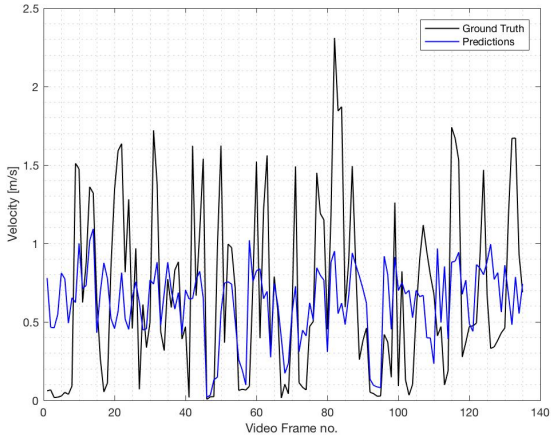

Figure 16. Predictions and Targets - training on UAV data

ror is reasonably low for the drone dataset, visual inspection shows that the predictions do not follow the general trend of the targets.

5. Conclusion

A deep learning model for the estimation of velocity from single static images is presented in this work. Since motion blur is proportional to the relative speed between camera and objects and there could be a subtle relationship between scene context and velocity. Analysis of the model using transfer learning show that the model learns features related to motion blur. However, it can be seen that the task of velocity estimation is also context dependent, given that the model was able to learn albeit with lower accuracy, even without motion blur when trained on vKITTI. Further experimentation shows that while the model does not adapt across domains, it is able to learn even when subject to changes in scene conditions. The limitations of the model are that it reduces in performance when subject to large illumination changes, does not have any domain adaptation capability.

Further experimentation could aid in forming a better understanding of the features learned by the model. One such experiment would be to constrain the scene context and vary the motion blur, by moving the camera to and fro in a linear fashion.

References

[1] K. Alexiev and I. Nikolova. An algorithm for error reducing in imu. In *Innovations in Intelligent Systems and Applications (INISTA), 2013 IEEE International Symposium on*, pages 1–6. IEEE, 2013.

[2] J. Andersh, A. Cherian, B. Mettler, and N. Papanikolopoulos. A vision based ensemble approach to velocity estimation for miniature rotorcraft. *Autonomous Robots*, 39(2):123–138, 2015.

[3] I. Biederman, R. J. Mezzanotte, and J. C. Rabinowitz. Scene perception: Detecting and judging objects undergoing relational violations. *Cognitive psychology*, 14(2):143–177, 1982.

[4] G. Cheng, Z. Li, X. Yao, L. Guo, and Z. Wei. Remote sensing image scene classification using bag of convolutional features. *IEEE Geoscience and Remote Sensing Letters*, 14(10):1735–1739, 2017.

[5] J. L. D. Sun, S. Roth and M. J. Black. Learning optical flow. In *European Conference on Computer Vision*, October 2008.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[7] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015.

[8] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.

[9] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2414–2423. IEEE, 2016.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[12] D. Gong, J. Yang, L. Liu, Y. Zhang, I. Reid, C. Shen, A. v. d. Hengel, and Q. Shi. From motion blur to motion flow: a deep learning solution for removing heterogeneous motion blur. *arXiv preprint arXiv:1612.02583*, 2016.

[13] V. Grabe, H. H. Bülthoff, and P. R. Giordano. On-board velocity estimation and closed-loop control of a quadrotor uav based on optical flow. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 491–497. IEEE, 2012.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[15] H. W. Ho, G. C. de Croon, and Q. Chu. Distance and velocity estimation using optical flow from a monocular camera. *International Journal of Micro Air Vehicles*, 9(3):198–208, 2017.

[16] D. Honegger, P. Greisen, L. Meier, P. Tanskanen, and M. Pollefeys. Real-time velocity estimation based on optical flow and disparity matching. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5177–5182. IEEE, 2012.

[17] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[18] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[19] M. B. Kjærgaard, H. Blunck, T. Godsk, T. Toftkjær, D. L. Christensen, and K. Grønbæk. Indoor positioning using gps revisited. In *International conference on pervasive computing*, pages 38–56. Springer, 2010.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[21] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] T. Lemaire, C. Berger, I.-K. Jung, and S. Lacroix. Vision-based slam: Stereo and monocular approaches. *International Journal of Computer Vision*, 74(3):343–364, 2007.

[24] H.-Y. Lin and K.-J. Li. Motion blur removal and its application to vehicle speed detection. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 5, pages 3407–3410. IEEE, 2004.

[25] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

[26] K. McGuire, G. de Croon, C. De Wagter, K. Tuyls, and H. Kappen. Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone. *IEEE Robotics and Automation Letters*, pages 1070–1076, 2017.

[27] T.-L. Pao and M.-D. Kuo. Estimation of the point spread function of a motion-blurred object from autocorrelation. pages 1226–1233, 04 1995.

[28] Pawit Kocakarn. Machine learning with python : Image classifier using vgg16 model - part 1: Theory.

[29] S. L. Pintea, J. C. van Gemert, and A. W. Smeulders. Déja vu. In *European Conference on Computer Vision*, pages 172–187. Springer, 2014.

[30] H. N. Ramakrishnan. *Detection and Estimation of Image Blur*. Master's thesis, Missouri University of Science and Technology, 2010.

[31] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1164–1172, 2015.

[32] S. Rezvankhah, A. A. Bagherzadeh, H. Moradi, and B. N. Araabi. A real-time velocity estimation using motion blur in air hockey. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1767–1772, Dec 2012.

[33] E. Santana and G. Hotz. Learning a driving simulator. *CoRR*, abs/1608.01230, 2016.

[34] A. Savakis and R. L. Easton. Blur identification based on higher order spectral nulls. 2302, 09 1994.

[35] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Vision-based state estimation for autonomous rotorcraft mavs in complex environments. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 1758–1764. IEEE, 2013.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[39] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *European conference on computer vision*, pages 140–153. Springer, 2010.

[40] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2443–2451. IEEE, 2015.

[41] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1385–1392. IEEE, 2013.

[42] S. Weiss, D. Scaramuzza, and R. Siegwart. Monocular-slambased navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

[43] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

[44] Y. Yitzhaky and N. Kopeika. Identification of blur parameters from motion blurred images. *Graphical Models and Image Processing*, 59(5):310 – 320, 1997.

[45] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[46] Y. Zhang, M. Bai, P. Kohli, S. Izadi, and J. Xiao. Deepcontext: Context-encoding neural pathways for 3d holistic scene understanding. *arXiv preprint arXiv:1603.04922*, 2016.

[47] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Preliminary Report

1

Introduction

Given the steady progress in the field of robotics and automation and its widespread acceptance and use in day to day life, it has become increasingly vital to estimate velocity for exploration, navigation and related tasks. With the advent of self-driving cars, autonomous unmanned aerial vehicles and other similar systems, it is now of paramount importance to obtain accurate velocity estimates of the agent in order to prevent accidents. There are many methods to measure the velocity of a given agent, such as using inertial measurement units, GPS (Global Positioning System), or laser-based sensors. However, errors accumulate in IMUs [1], and thus they need a supplementary system such as GPS for correction. Also, GPS as a standalone system has poor performance in some indoor scenarios [2]. However, cameras are cost-effective, compact and low power while still providing robust and reliable data [3]. Thus, in recent times drawing inspiration from examples in nature, several computer vision based approached have been put forth for velocity estimation [4] [5][6].
The handling of visual data in an effective manner was made easier with the introduction of convolutional neural networks [7] [8]. Following the seminal work of Alex Kriezehvsky et. al in 2012 [9] on the IMAGENET object recognition challenge [10], convolutional neural networks gained popularity. Using GPU acceleration made deep neural networks computationally more tractable. Convolutional neural networks have since been used to learn how to predict optical flow [11] with significant accuracy. Convolutional neural networks have also been used for heterogeneous blur removal [12] and learning to classify blur kernels [13]. Research has also been carried out to estimate velocity from blur present in images [14]. Thus it is possible that motion blur contains velocity information and blur kernel identification can be achieved with a learning based framework.

This work aims to investigate the possibility of using machine learning techniques to directly estimate velocity from an image. Towards that end, this literature review is carried out to present a critical analysis of the state of the art in the topic of estimating velocity from images and also to provide a deeper understanding of possible features contributing to the mapping of images and their associated ego-velocity.

In this introductory chapter, section 1.1 presents the motivations behind that drive this body of research, the section following that outlines the research questions and the sub-questions to be addressed while working towards the research objective. The final section of this chapter discusses the experimental setup and facilities to be used while working on this project.

The rest of the report is organized as follows, the second chapter presents the necessary theoretical background on relevant machine learning algorithms. Since motion blur has been proven to contain velocity information [14], the third chapter discusses motion blur and how various approaches have leveraged it for velocity estimation. The fourth chapter presents the preliminary results obtained.The third and fourth chapter provide insight as to how velocity can be estimated from single images. The final chapter presents the conclusions drawn from the literature study.

1.1. Motivation for Research

State estimation poses a challenge when working towards robot autonomy, since the robot needs to be aware of its own parameters to interact with the environment in a defined manner. The position of the agent can be estimated with sensor fusion of Inertial Measurement Unit(IMU) data with GPS or magnetometer measurements. Since sensor bias increases in case of IMUs, accurate non-linear dynamic models or fusing input from other sensors are required. Velocity estimation has also been achieved by fusion of IMU data with visual data [15]. In GPS denied environments, laser range or sonar sensors are alternative solutions. Another popular sensor option for state estimation are RGB cameras, because of the richness of features and information in an image stream.

Cameras as sensors have been used in a wide range of applications especially on board unmanned aerial vehicles. These applications include obstacle avoidance, state estimation [16], integration with an inertial navigation system [17] and navigation [18]. Another one of the main advantages of cameras as sensors apart from being information rich is that they can function independent of the platform/agent that they are deployed on. For instance, state estimation (using potentiometers) in humanoid agents is quite different from that on board unmanned aerial vehicles(IMU), however both agents can utilize vision based state estimation.

Research in computer vision has already previously investigated motion estimation from single static images [19] via structured regression with a random forest. This thesis builds on the previous work to analyze what kind of role motion blur and context play, in the prediction of motion from single images. Towards this end, machine learning models are trained to estimate velocity from single static images and then analyzed to determine what the model learns.

One of the current methods of vision based velocity estimation is optical flow computation between two successive image frames. The storage of the previous frame requires memory which can potentially be saved by using single images.This is particularly advantageous when working with embedded systems. There are methods of optical flow based estimation that do not require dense computations [5], however these methods still require storing optical flow data(albeit down-sampled) from the previous frame. Velocity estimation techniques based only on single images, would thus require lesser memory, making it an interesting topic to investigate. Also it can be seen in literature that velocity can be estimated with information from a single motion blurred image [20]. Since blur parameters can be learned and motion blur has been shown to contain velocity information, using a learning based framework for velocity estimation would be of interest to investigate.

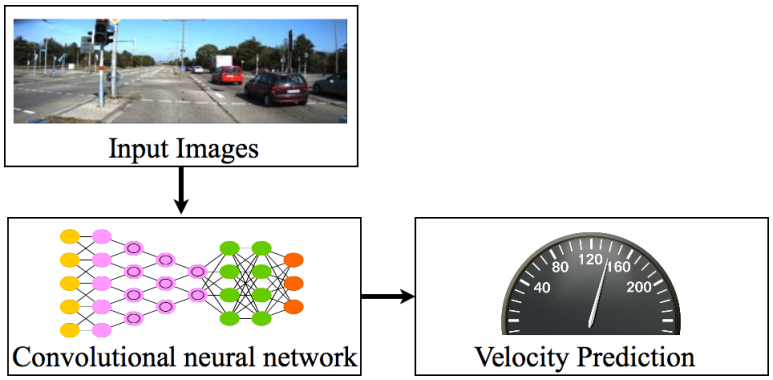

Figure 1.1: Velocity estimation pipeline

Some machine learning models such as deep learning are notorious for requiring massive amounts of computational power and data to train. The emergence of general purpose GPU computing and the parallelization prowess of GPUs have reduced training and deployment times for deep learning techniques. With recent advances in embedded deep learning, model inference is now easily achieved on mobile GPUs [21]. Thus it is significant to investigate utilizing deep learning for velocity estimation from single images particularly for drone based applications.

1.2. Research objective and research questions

The overarching objective of this research project is to explore the possibility of determining the velocity from single image frames. From the research objective, the following research question/sub-questions are derived,

- *How can a model be developed that can learn to estimate motion from a single static image?*
  - *Which kind of learning based model would be suited for the task?*
  - *What are the features that could enable the model to correlate image data and ground truth velocity?*
    - *What is the contribution of each feature/cue toward learning the mapping from pixel space to velocity?*

1.3. Pre-requisites & Facilities

The training of many learning based models such as deep neural networks are computationally expensive. Therefore owing to the parallel computation power of GPUs, GPU acceleration has become quite popular. For this body of work the DAS4 cluster at TU Delft is used [22] which has an NVIDIA GTX 1080 GPU at the node being utilized. The prospect of deploying the velocity estimation model on data collected from a drone is also to be investigated, for which a Parrot bebop drone will be used.

The training and deployment of learning based models could potentially require developing a lot of programs to facilitate individual parts of the model such as optimization, model architecture generation etc. In order to avoid reinventing the wheel, pre-existing libraries are made use of. Recently, quite a few libraries for learning based models such as Tensorflow, keras, Theano, PyTorch and caffe have been developed. Caffe [23] is a framework specialized for development and deployment of models implemented with C++ and Google protocol buffers for designing network architectures and controlling their training/testing. It also provides APIs for Python and Matlab. The Caffe library provides a clean architecture, great performance and detailed documentation. The network architecture is defined separate from the solver thus also offering modularity and flexibility. Furthermore, Caffe also has a 'Caffe Model Zoo' where models can be obtained and shared along with their definitions and trained weights, thus allowing researchers to build on top of each other's work. Thus, Caffe was the ideal choice for the implementation of this work.

2

Background on Neural Networks

This chapter presents the theoretical background necessary to gain a deeper understanding of this work. Over the past couple of decades, with the increasing amount of data and computational power available, machine learning has become prevalent in today's world. Machine learning can be broadly categorized into three paradigms, which are supervised, unsupervised and reinforcement learning algorithms.

Supervised learning involves data samples with targets or labels which provide 'supervision' for the model to learn a mapping function between the input data and its corresponding target/label. The approximated mapping function can then be used to predict the labels or targets of unseen input data. Supervised learning problems can again be categorized as either classification or regression. In classification, the output corresponds to probability distribution across predefined classes. In regression problems, the output is a real value based on input.

Unsupervised learning works with input data that has no corresponding predefined targets or labels. These algorithms focus on learning the underlying structure or distribution of the data, such as clustering where inherent groupings of the data are discovered.

Reinforcement learning draws inspiration from behavioural psychology. An agent(physical or simulated) receives a 'reward' based on the state and the action it takes upon the environment. Rather than training on predefined correct input output pairs, reinforcement learning focuses on online performance, balancing the trade-off between exploring the state/action space and exploiting the knowledge it gained.

Given that the task at hand is to estimate velocities from images, the continuous real valued output implies that supervised learning and in particular regression would be an appropriate choice of model. However, Even within the supervised learning paradigm, there are many approaches and algorithms such as random forests, support vector machines, neural networks and so on. Support vector machines have one caveat, which is that an optimal model requires prerequisite knowledge of the data at hand in order to choose an appropriate kernel. Whereas methods like random forests and neural networks are able to automatically determine the structure in the data. However if the data is sparsely distributed, or if the data is not axis aligned, random forests might not perform well. Neural Networks, on the other hand are versatile and capable of approximating complicated functions.

The subsequent section and subsections of the chapter presents the principle behind neural networks and also discusses more sophisticated variants such as recurrent neural networks and convolutional neural networks.

2.1. Artificial Neural Networks

Artificial neural networks(ANNs) draw inspiration from biological neural networks of the brain. McCulloch and Pitts created the basis for modern day neural networks by defining a computational model called 'threshold logic '[24].The artificial neural networks are a collection of connected computational nodes that iteratively improve their performance in tasks using examples rather than explicit task specific programming. The standard feed forward neural network consists of an input layer, an output layer and a hidden layer. The input layer propagates the data to the hidden layer which in turn is connected to the output layer. Each computational node is connected to all the nodes from the previous layer. Each connection of the network has a weight associated with it. The weights are progressively tuned until they represent a mapping between the inputs and the outputs.Figure 2.1 represents the working of a single neuron, it first performs a sum operation on the weighted inputs adding any bias and then applies a non-linear activation function.

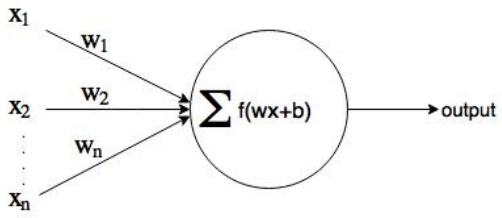

Figure 2.1: A single neuron, inputs x are multiplied by weights w and bias values are added, this resulting value is then subject to an activation function.

The activation function can be defined to be the mathematical abstraction which represents the firing rate in biological neural networks present in animal brains. It decides whether a neuron 'fires' or not. The value of the weighted input and bias can be any real number, the activation function subjects it to specific bounds. The most common activation functions are the sigmoid, hyperbolic tangent and the rectified linear unit (ReLU) as shown in figure 2.2 below. The ReLU activation function has been shown to be six times faster than an equivalent network with hyperbolic tangent as its neuron activation function[9].

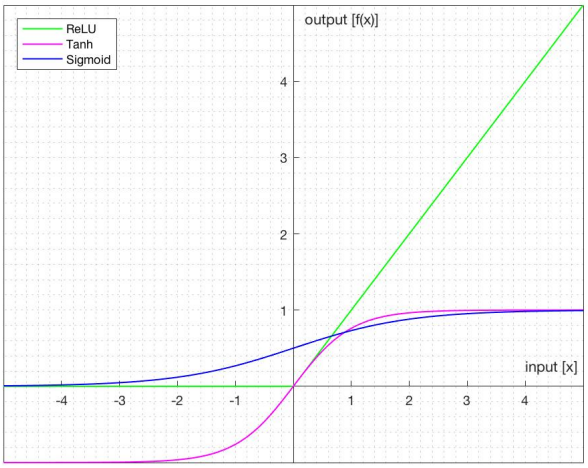

Figure 2.2: Common activation functions

The artificial neural network model operates in two different phases, a training phase and a testing or deployment phase. In the training phase, the network iteratively tunes its weights such that it produces the correct output when presented a sample. During this training phase the neural network is supplied

with a set of input vectors $x_n = 1, 2, ..., n$ and the corresponding set of correctly labelled outputs y_n. The goal is to minimize the error between the output predicted (y_p) by the network and the label as measured by the cost function by adjusting the weights. A common cost function used is the sum of squared errors as given below.

$$J = \frac{1}{2}\sum_{n=1}^{N} ||y_p - y_n||^2 \tag{2.1}$$

To minimize the cost function, it can be treated as an optimization problem, thus its slope with respect to each weight of the network(The gradient) is to be calculated. During each iteration the weights are adjusted in the direction that produces the steepest descent along the error space. This direction is determined by computing the negated gradient of the cost function with respect to each of the weights.

$$\nabla J(\omega_n) = \left[\frac{\partial J}{\partial \omega_0}, \frac{\partial J}{\partial \omega_1}, \frac{\partial J}{\partial \omega_2}, ... \frac{\partial J}{\partial \omega_n}\right] \tag{2.2}$$

The weights are then updated with the following rule,

$$\omega_k = \omega_k - \alpha \nabla J(\omega_k) \qquad k = 0, 1, 2, ...n \tag{2.3}$$

Where α is the learning rate parameter which determines the rate at which the parameters are updated. The gradients for a feed forward neural network are computed with the backpropagation algorithm as summarized below,

Algorithm 1 Backpropagation algorithm

Initialize the weights of the network
While: max. Iterations > Iterations completed (or other stopping criterion)
  for every input/output data pair or batch of data **do**
    Forward Pass
      Propagate the input forward through the network.
      for Every hidden or output layer **do**
        for Every neuron j in the layer **do**
          Compute sum of input with respect to each neuron of previous layer $I_j = \sum_i w_{ij} O_j + \theta_j$
          Compute activation of each unit $O_j = f(I_j)$
        end for
      end for
    Backward Pass
      for each unit j in the output layer **do**
        Compute the error $Error_J = O_j(1 - O_j)(T_j - O_j)$
      end for
      for each unit j in the hidden layers **do**
        Compute error with respect to previous layer k $Error_J = O_j(1 - O_j)\sum_k Error_k w_{jk}$
      end for
      for Each weight in the network **do**
        $\Delta w_{ij} = -O_i Error_j \alpha$
        $w_{ij} = w_{ij} + \Delta w_{ij}$
      end for
      for Each bias in the network **do**
        $\Delta b_{ij} = -Error_j \alpha$
        $b_{ij} = b_{ij} + \Delta b_{ij}$
      end for
  end for

Backpropagation is essentially a generalization of the chain rule applied to all layers of the network where the rule is used to compute gradients at each layer. Backpropagation has two stages, the forward pass, where the input vector is processed by all layers of the network and produces the output evaluated by the cost function and a backward pass which is when the chain rule is applied and the parameters are updated. Backpropagation can be used in an online or offline setting. Offline or batch learning is faster for relatively smaller datasets but is prone to getting stuck in a local minimum. Online learning performs better and has the ability to adapt to the unseen data better [25].

The regular neural networks do not scale well to multi-dimensional input such as images due to the full connectivity, they suffer from curse of dimensionality. Assuming an image of size 32 x 32 x 3 then the neuron of the first hidden layer would have 3072 weights. With several such neurons parameters would increase which might lead to overfitting. Thus, convolutional neural networks are more suitable to this task as they constrain their architecture in a more sensible way to handle multi-dimensional data.

2.1.1. Convolutional Neural Networks

Convolutional neural networks are a class of deep, feed forward neural networks inspired by connectivity patterns of the animal visual cortex arising as a result of work done at Bell labs [7] [8]. Conventional neural networks such as the multi-layer perceptron accept a vector input which they transform based on transfer functions of their hidden layer, weights and biases. CNNs in contrast, take the full raw images as input and employ 3D volumes of neurons for learning feature maps. The 3D volumes are followed by the fully connected layers which discard the structural information inside the image. CNNs were introduced to reduce the amount of parameters to tune by instituting a new connectivity pattern between the neurons inspired by the organization of the cat's visual cortex. The individual cells of the visual cortex are arranged in such a way that they are only sensitive to certain sub-regions of the visual field, called a receptive field. These sub-regions are then tiled to cover the entire visual field and the cells act as local filters over the input space. Despite being very different from the multi-layer perceptron, convolutional neural networks are also trained using backpropagation.

Convolutional neural networks typically consist of three types of layers: convolutional layers, pooling layers and fully connected layers.There are however, variations of convolutional neural networks such as fully convolutional networks which do not have fully connected layers. Understanding the principle behind the working each of the aforementioned layers will provide insight as to how information is encoded in the network. The input layer is a three dimensional volume corresponding to the the dimensions of the input provided to the network. The successive layers perform various operations on this input.

Convolutional layer

The convolutional layer is a crucial building block of a convolutional neural network. To better understand the convolutional layer, the convolution operation is first discussed. It is assumed that pixels that are spatially closer together would combine to form a feature of interest rather than pixels that are farther apart. Given a two-dimensional image, I, and a small matrix, K of size h × w, (known as a convolution kernel), the convolved image is computed by sliding the kernel over the image in all possible ways, and determining the sum of element-wise products between the image and the kernel:

$$(I * k)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij}.I_{x+i-1,y+j-1} \tag{2.4}$$

In addition to parameters of the layer, Convolutional layers also have 'hyperparameters'. The output of a convolutional layer is defined by these hyperparameters. The receptive field which is essentially the size of the filter is one hyperparameter. The depth of the output volume is another hyperparameter, and it corresponds to the number of filters. Stride and zero padding are the other two hyperparameters.The stride specifies how many pixels are skipped as the filter slides over the image. A value of one means the filter is moved one pixel at a time. Zero padding is the number of zeros added to the input volume along the border. A parameter/weight sharing scheme is also employed in convolutional neural networks. It is assumed that a feature that is useful to compute at one spatial location is also

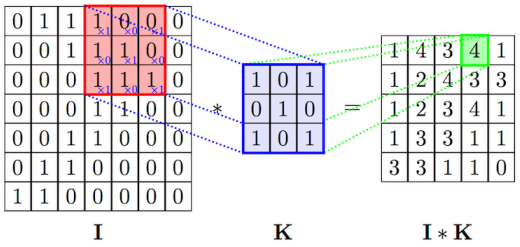

Figure 2.3: Example of convolution operation. [26]

useful at other spatial locations. Thus, a single two-dimensional depth slice (assuming volume of 10 x 10 x 15, there are 15 10 x 10 depth slices) is constrained to have the same weights and bias.

For the forward pass, the filters in each convolutional layer 'slides' across the inputs width and height performing a dot product between each element in the filter and the image(convolution). This produces an activation/feature map which is essentially a 2D matrix. The network 'learns' these filters by backpropagation of errors, such that the filter generates an activation response when it detects some specific type of feature at a position in the input image. When these feature maps are stacked along the depth, the output volume of the convolutional layer is obtained.

Pooling layer

After each convolutional layer, there is usually a pooling layer. The function of a pooling layer is to progressively reduce the spatial size of the representation of the input, thus reducing the amount of parameters of the network and ultimately preventing overfitting. Pooling layers typically perform a MAX operation or an averaging operation independently per depth slice of the input. In case of MAX pooling, a window of fixed size is slid over the input volume and the maximum value of each window is taken as the corresponding part of the output volume. Average pooling involves averaging the value in each window and the average value is taken as the output.

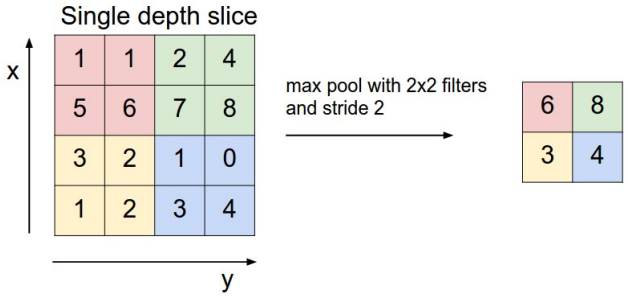

Figure 2.4: Example of Pooling operation. [27]

Fully Connected layer

The final part of a standard convolutional neural network are fully connected layers. The fully connected layers are operate in a manner similar to regular feed forward networks.
To summarize, convolutional neural networks exploit spatial location information by means of a localized connectivity pattern between nodes in adjacent layers. This way, the filters that are learned produce a higher activation response to the localized input pattern. When these layers are stacked, they become hierarchically more non-linear and thus responsive to a larger region of the input image. The network first creates representations of small parts of the input, then from them, it assembles representations of larger areas of the input. Every single filter is repeated across the whole visual field. These filter units share the same parameters (weight vector and bias) and thus form a activation/feature map of that layer. So it can be said that all the neurons in a given convolutional layer will 'respond' to the same feature (within their specific response field). Repetition of units in such a manner allows for the features to be detected irrespective of their spatial position in the visual field, thus enabling the

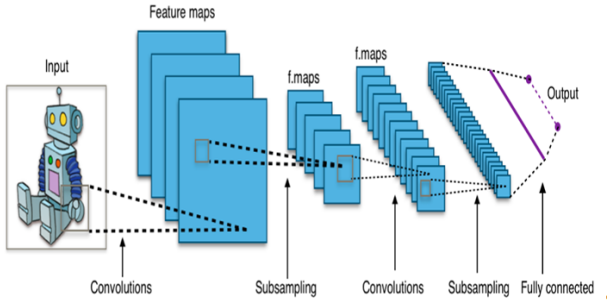

Figure 2.5: Example of convolutional neural network architecture. [28]

property of translation invariance. The pooling layers perform down-sampling, and the fully connected layers establish high-level reasoning.

2.1.2. Recurrent Neural Networks

Recurrent neural networks are different from regular feed forward networks in the sense that they have at least one feedback loop, that is the input is not just the current sample but also what the network had perceived in previous time steps. Since the activations can flow in a loop, the network can do temporal processing and learn sequences. The sequential information is preserved in the recurrent network's hidden state, which could span multiple time steps as it cascades forward affecting the processing of every new example. Thus, it finds correlations between events separated in time. Mathematically the hidden state can be defined as,

$$h_t = \phi(Wx_t + Uh_{t-1}) \tag{2.5}$$

Thus, the hidden state is a function of the input at a given current time step x_t, multiplied by a weight matrix W and added to the hidden state of the previous time step $h_t - 1$ which is multiplied by a transition matrix, similar to a Markov chain. The weight matrices determine how much importance the present input and the past have.

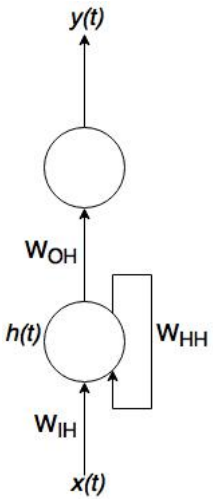

Figure 2.6: A recurrent neural network

The recurrent neural network can also be thought of as a dynamic system represented by the state space equations,

$$\begin{aligned} h(t) &= f_h(W_{IH}x(t) + W_{HH}h(t-1)) \\ y(t) &= f_o(W_{OH}h(t)) \end{aligned} \tag{2.6}$$

Where the matrices W_{IH}, W_{HH}, W_{OH} are the weights of the layers and f_h, f_o represent the activation functions of the hidden and output layers respectively. A more intuitive explanation for recurrent neural networks is that they are feed forward neural networks when they are unfolded or rolled out over time as shown in Fig.2.7. Therefore, the concepts used for feed forward neural networks are generally applicable here.

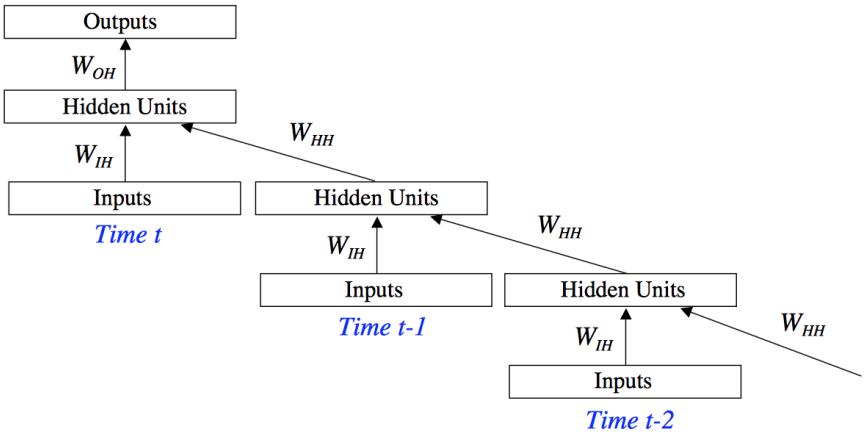

Figure 2.7: Recurrent network unfolded over time. [29]

An extension of the backpropagation algorithm known as the backpropagation through time (performing gradient descent on a network unrolled through time) is used to train recurrent neural networks. In backpropagation through time, since a whole sequence is used as a training sample, the error needs to be summed up across all time steps.

$$E(y_p, y_t) = \sum_t E_t(y_p, y_t) \tag{2.7}$$

Similarly the gradients are also summed up across all time steps.

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \tag{2.8}$$

The gradients are calculated using the chain rule. If the third time step is considered, the gradient of weights representing the outer and hidden layer connections are given by,

$$\begin{aligned} \frac{\partial E_3}{\partial W_{OH}} &= \frac{\partial E_3}{\partial y_{3_p}} \frac{\partial y_{3_p}}{\partial W_{OH}} \\ &= (y_{3_p} - y_{3_t}) \otimes h_3 \end{aligned}$$

Thus, W_{OH} depends on values from the current time step. But when considering W_{HH} and W_{IH},

$$\frac{\partial E_3}{\partial W_{HH}} = \frac{\partial E_3}{\partial y_{3_p}} \frac{\partial y_{3_p}}{\partial h_3} \frac{\partial h_3}{\partial W_{HH}} \tag{2.9}$$

Here, h_3 is not a constant, rather $h_3 = f_h(W_{IH} x_3 + W_{HH} h_2)$ thus it depends on h_2 which in turn depends on h_1. Thus, the gradient can be calculated by,

$$\frac{\partial E_3}{\partial W_{HH}} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial y_{3_p}} \frac{\partial y_{3_p}}{\partial h_3} \frac{\partial h_3}{\partial h_k} \frac{\partial h_k}{\partial W_{HH}}$$

The update rules for backpropagation through time are similar to that of standard backpropagation.

Since the layers and time steps of deep neural networks are related to each other in a multiplicative manner, derivatives are susceptible to vanishing or exploding.The gradient contains information as to how the weights needed to be updated for minimum error. Updates to the parameters cannot be achieved without knowledge of the gradients, which is where recurrent neural networks struggle, learning long-range dependencies between inputs that are several time steps apart. This can be seen in equation 3.9, where h_3 represents a chain rule in itself since it is dependent on h_2 and h_1. Thus, the differentiation of h_k with respect to a previous layer is the derivative of a vector function with respect to a vector, the resulting matrix is a jacobian, where the elements are all the point-wise derivatives. This is akin to the effects of applying a sigmoid function over and over again(Fig.2.8). The data is flattened until it has no detectable slope when considering large stretches. This is similar to a gradient vanishing as it passes through multiple layers.

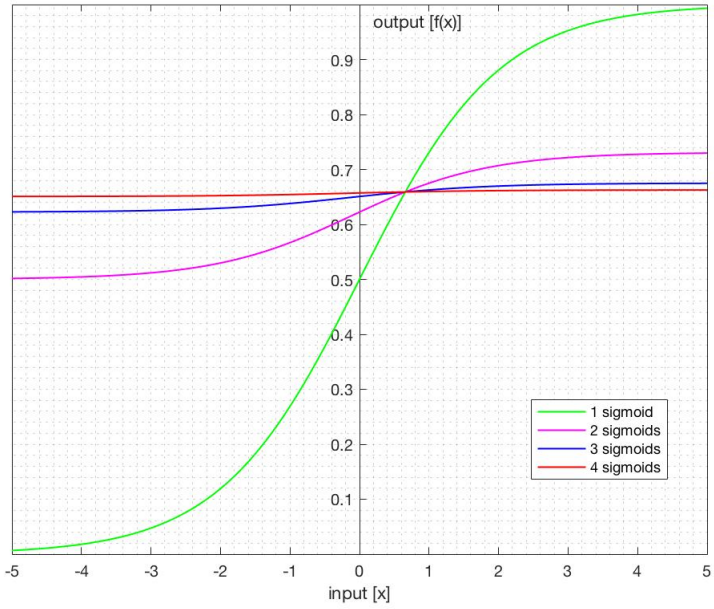

Figure 2.8: Vanishing gradient due to multiple applications of sigmoid function

In some cases, the activation function can be chosen such that the gradients increase exponentially. In case the gradient becomes too large, a truncated form of the backpropagation through time can be utilized. The disadvantage is that the gradient has a limited ability to flow back due to truncation, therefore the network can't learn dependencies that are stretched far apart in time.

Long Short term Memory

The LSTM is a recurrent neural network architecture proposed by Sepp Hochreiter and Jürgen schmidhuber in 1997 as a solution to the vanishing gradient problem. LSTMs aid in the preservation of error that can be backpropagated through both time and layers of the network. By maintaining a more constant error, LSTM allows recurrent neural networks to learn over many time steps, thereby instituting a channel to link causes and effects. LSTMs introduce a cell state, (denoted by C_t in Fig.2.9) which flows along the unit like a conveyor belt with a few interactions with data. Information is added or removed from gates by means of structures known as gates. The gates (represented by σ in Fig.2.9) 'regulate' the information passing through them. The gates are sigmoid neural network layers which output a lower value if the information should not affect cell state or higher to affect the cell state.

The forget gate is represented by f_t and f_{t-1} in Fig.2.9. The forget gate regulates the contribution of previous cell state to the current cell state. It takes the previous time step activation and current information as input, and outputs a number between 0 and 1 for each number in the cell state Ct−1.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$

The next step in an LSTM model is to which new information is to be stored in the cell state. A sigmoid layer called the input gate layer decides which values are to be updated. Then, a hyperbolic tangent layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. The two values are combined to create an update to the state.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

The last stage determines the output of the single unit in the LSTM.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

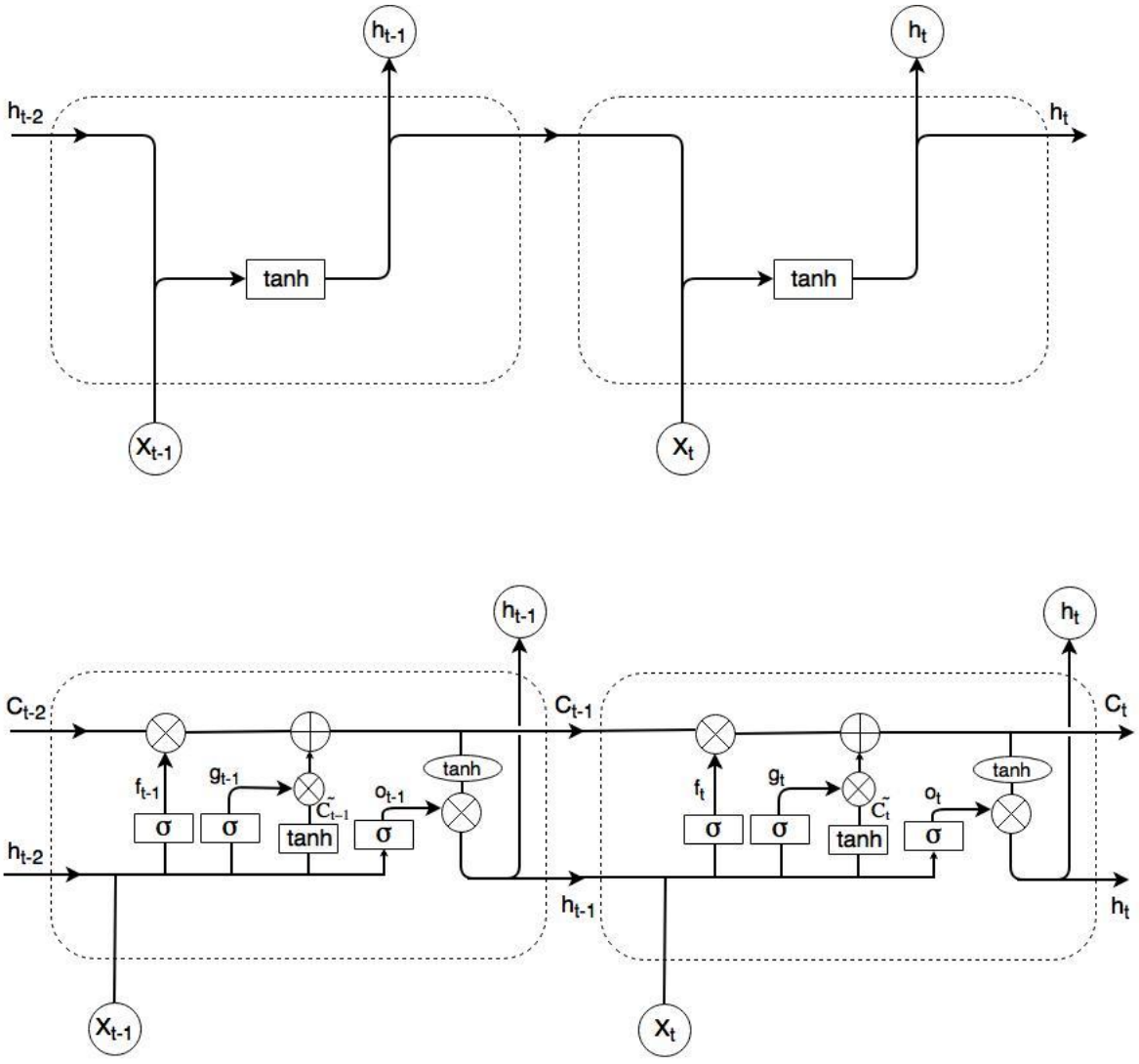

Figure 2.9: Standard RNN (top) vs LSTM (bottom)

From the theory presented in this chapter, it is evident that convolutional neural networks are best suited for handling image data since the weight sharing mechanism ensures there are lesser parameters and they have shown promising results in a wide array of image based tasks like classification and localization [9], depth estimation [30] and many more. Convolutional neural networks are thus the learning based model that will be further investigated for the task of image based velocity estimation.

3

Motion Blur & Motion Estimation

When a camera is subject to motion within an exposure period, the illumination changes are integrated over time and the sharpness is smeared over the image, thus forming motion blur. In other words a captured image can be thought of as an averaged sample over a time period, as a result, moving objects in that time period cause motion blur. Image deblurring to recover the original image is an actively researched topic since traditional computer vision tasks such as segmentation and tracking are difficult to perform without knowing the blur kernel. However, the motion blur being removed actually has motion information. Thus, it is possible that a model could learn a function that maps motion blur parameters to velocity of the camera.

Given the characteristics of a particular image, geometric theory can be used to solve the object velocity measurement problem. The advantages of this method are that it is simple in theory. However, the disadvantages are that it requires an additional reference object, including length information, and it exhibits a significant estimation error. Another popular method for velocity estimation given a sequence of images is using optical flow. Motion estimation here is the estimation of the displacement and velocity of features in an image frame with respect to the previous frame in a time sequence of images.

This chapter is organized into three sections. The first section presents mathematical formalism for motion blur and its characteristics to provide insight as to how velocity can be estimated from it. The second section discusses the related body of work involving motion estimation from images. The final section is regarding how the context in images influences perception based tasks and the related research work that has exploited this correlation. The final section is presented to provide a basis to analyze the role of context in velocity estimation.

3.1. Characterization of Motion Blur

An image with motion blur retains information that parameterizes the blur. This enables the recovery of motion from a single static image. Motion blur in an image can be characterized by its Point Spread Function(PSF). A motion blurred image (b) can be thought of as a convolution of the unblurred image(i) and the point spread function(h) with additional noise(μ).

$$b(x,y) = i(x,y) * h(x,y) + \mu(x,y) \tag{3.1}$$

The point spread function is defined as:

$$h(x,y) = \begin{cases} \frac{1}{L}, & \text{if } \sqrt{x^2+y^2} \leq \frac{L}{2} \text{ and } \frac{x}{y} = -tan(\phi). \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

Where ' L ' is the motion length and ϕ is the motion direction. In order to recover and infer information from the motion blur, accurate estimates of the motion blur PSF parameters, length and motion angle need to be computed. From the motion blur parameters, relative velocity can be recovered if the exposure time is known.

The recovery of the point spread function is a particularly challenging task owing to the short duration of the image degradation process and loss of information it causes. However, study of motion blur PSF has revealed that it exhibits a unique behavior in the frequency domain [31]. Blur identification with spectral nulls making use of power spectrum and the power cepstrum was investigated [32], but this approach fails when there is a large amount of noise or if the blur size is small.

A considerable amount of research has been done with regards to autocorrelation based methods for motion blur PSF estimation. One approach evaluates the phase gradient of the image to extract the blurred edge [33]. With this extracted part, an autocorrelation matrix is determined and its contour is drawn. The blur direction is obtained from the trace of the peak of the autocorrelation matrix, also a first order backward difference equation is evaluated on the trace of the peak to obtain blur extent. This approach is impervious to noise and retains motion characteristics.

Another method based on autocorrelation for PSF estimation is first determining the motion direction and then the pixel blur [34]. An approximation of the image derivative in a specific direction is first obtained and then the total image intensity in the same direction is computed by the summation of the absolute values of all pixels. The blur direction is then defined as the direction in which the total intensity of the absolute values of the image derivative is lowest. A digital autocorrelation function (ACF) to the image derivative lines in the motion direction identified above is then calculated and the average of the ACF is obtained. The relative distance between the center point and the minimum of the averaged ACF gives the extent of the blur in terms of number of pixels. However, the article mentions that this method works well only if the relative homogeneity(ratio of minimum difference between the values of the pixels at endpoints of the smear track of the point and the adjacent pixels outside the track to the maximum difference between two adjacent pixels inside the track) of the PSF is larger than unity. Also, any error in estimating the motion direction affects the calculation of blur extent.

A motion blur parameter estimation was based on frequency response was proposed in [35] which uses a 2D Gabor filter to estimate the angle and a neural network is trained to obtain the blur length. The Gabor filter is convolved with the log spectrum of the blurred image at different orientations to obtain the response corresponding to different frequencies and orientations. The L2 norm of the convolved image is calculated for each orientation. The angle corresponding to the highest L2 norm gives the blur direction. A Radial basis function based neural network was trained to obtain the blur length. The key issue with most of the aforementioned methods is that they all require incorporation of domain knowledge in the framework, engineered features or pre-processing.

Recent research in deep learning has shown that, the motion blur parameters can be estimated from raw images itself rather than resorting to frequency domain analysis. Convolutional neural networks have particularly been used for blur identification [13]. Although the article describes a method for motion blur removal, the first part of the approach focuses on identification of blur parameters. The motion space is discretized and the convolutional neural network is trained as a classification model to estimate probabilities of motion kernels for each patch of the image, then dense motion blur kernels for the whole image is estimated using a Markov random field.

Fully convolutional neural networks have also been used to estimate motion flow from motion blur in an image [12]. To estimate the blur kernel, a generic and information rich prior would be necessary. Rather than learning a prior over the latent image, which would require modelling all the image content, motion flow is learned by the fully convolutional network thereby allowing the model to focus on the cause of blur irrespective of the contents of the image. This approach transcends patch level learning and does not require any post processing unlike the previously mentioned method. Training and testing of the model is directly performed on the whole image, thus exploiting additional spatial information and allowing it to estimate a dense motion flow map accurately, as seen in the Fig 3.1. However, here for motion flow estimation, the fully convolutional network is trained over discrete outputs. Discretization of the motion space could pose an issue(loss of information) as motion is an inherently continuous quantity.

From the literature that has been studied, the inference that can be drawn is using the motion blur

(a) Blurred Image (b) Xu and Jia (c) Sun et. al (d) Gong et. al

Figure 3.1: Motion Flow Estimation for Deblurring [12]

angle and motion blur length for the characterization of the point spread function and ultimately the motion blur has proved to be a successful approach. Machine learning techniques have been used to approximate mappings between pixel/feature space to the motion space, the body of research involving subsequent estimation of velocity from motion blur in images is presented in the following section.

3.2. Velocity and Motion Estimation from images

Research on image based motion estimation has increased in recent years[14][20], since computational power has become available to process the large amount of data generated by images at low latency and in real time. The aforementioned methods rely on estimating the motion blur parameters and then using the pinhole camera model to establish a relationship between motion blur and velocity. The predominant method for velocity estimation from images is by using optical flow which was first introduced by an American psychologist named James Gibson . Optic flow is the apparent motion of texture in the visual field relative to the camera. This information tells us not only how fast the camera moves and how close it is relative to the objects it sees. However, the computed optical flow field does not directly offer information on distance to surfaces or the egomotion velocity of the camera but it can be computed with additional input, such as control input to the agent [36].

When a stereo camera system is used [5], the optical flow measurements can be scaled with the distance between camera and the observed scene to obtain metric velocity of the agent. The optical flow field is the projection of the 3-D velocity field on the image plane. This flow field is then expressed as a sum of its translational and rotational components. The metric velocity is then obtained by transforming the translational velocity using the inter-axial distance between the two cameras and the disparity value estimates.

Monocular vision systems have also been utilized for optical flow estimation. H.W.Ho et al. [36] estimates the distance to the ground and the vertical velocity of an unmanned aerial vehicle. An extended Kalman filter was then used to obtain the height and velocity based on control input and flow divergence information. Grabe et al. [6] devised and implemented closed-loop control and onboard velocity estimation on a UAV. This was achieved using an algorithm for self motion estimation via optical flow extraction. The work builds upon the usage of the continuous homography constraint for egomotion estimation. Variations of the classical 4 point algorithm were used to compute the continuous homography matrix which encodes both the camera linear and angular velocity, and the scene structure.

Since the initial methods proposed by Horn & Schunck [38], Lucas & Kanade [39] most of the traditional approaches to estimate optic flow have focused on variational methods and in recent times, extensions to these methods involving large displacements and combinatorial matching have been proposed. One such example is the work termed ´Deepflow´ [40] which proposes a novel descriptor matching algorithm tailored for the optical flow estimation problem alongside a variational approach, quasi-dense correspondences are extracted and fed as input to an energy minimization framework. Another similar approach called ´EpicFlow´ [41] again proposes a two step method. The first step of the algorithm involves the sparse to dense interpolation of matches while preserving the edges. To this end, an edge aware geodesic distance measure which can handle motion boundaries and occlusion is designed. The dense matches obtained from the interpolation are then subject to the variational energy minimization framework to obtain the optical flow estimation. The disadvantage to these methods are that they

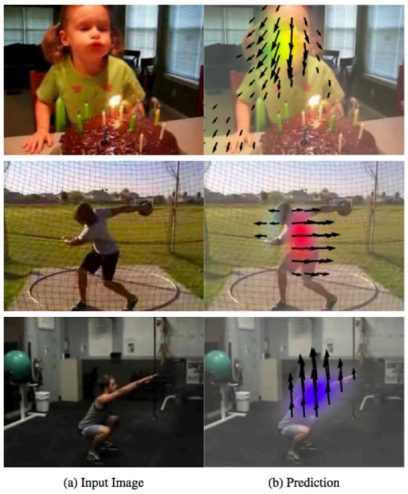

Figure 3.2: Optical flow prediction from static image [37]

require engineered methods for matching, aggregation and interpolation.

A significant amount of research has been carried out for the estimation of optic flow using machine learning algorithms. An early approach to learning optical flow was put forth by Black et. al [42] propose using Principal Component Analysis(PCA) to learn a set of basis flows. Flow fields are then estimated as a linear combination of these basis flows, Image derivatives are used in a gradient based framework to compute the coefficients of the linear combination. One key drawback of this method is that it assumes that motion occurring in images will be a linear combination of learned flows, which need not necessarily be the case. It is possible that the basis flows do not capture all the information in the motion space or a non-linear combination of basis flows could be present in the image.

Sun et. al [43] analyze the underlying statistics of optical flow, thereby proposing a steerable random field to model the statistical relationship between image and flow boundaries. Unsupervised learning has also recently been employed for the optical flow estimation problem. Taylor et. al [44] proposed a convolutional restricted Boltzmann machine capable of extracting low-level motion features which is ultimately used for action recognition. Konda et. al [45] propose a model to learn depth and motion using an energy based model, the synchrony autoencoder. The common issue with these approaches are that they work with controlled experimental setup and do not have performance comparable to classical flow estimation algorithms on real-world data.

The neural network architecture termed 'FlowNet' put forth by Fischer et. al [11] is capable of learning to predict optical flow with good generalization capability. The architecture proposes a convolutional neural network trained end to end to compress information spatially and then refined to obtain optical flow prediction. Ilg et. al [46] further built upon 'FlowNet', proposing a new framework 'FlowNet 2.0' which includes a stacked architecture warping the subsequent frame in the image with the intermediate optical flow prediction. It is also augmented with a sub-network used to specifically handle small displacements.

On a related note, Deep learning research has also recently shown that it is possible to extract a sequence of sharp images from a single blurred one. Purohit et. al [47]propose the training of a recurrent autoencoder with convolutional LSTM modules. Thus motion representations are learned in an unsupervised manner and ultimately video reconstruction serves as the surrogate task for training facilitating blur to video generation. Another approach introduced by Jin et. al [48] makes use of convolutional neural networks with larger receptive fields to combat the ill posed problem of deblurring an image. This work also introduces a novel loss function invariant to the temporal ordering of image frames.

Thus, it can be inferred from the literature that optical flow can be used to estimate velocities. With just a single static image, research has found that convolutional neural networks can be used for dense optical flow prediction [37] as seen in image 3.2. The convolutional neural network can predict motion in terms of optic flow as shown in the image above. From the literature, it is evident that single static images can be used for optical flow prediction and it is also seen that optical flow can be used for velocity estimation. This forms the core idea for this research work.

3.3. Role of Context in Perception

As human beings, we unwittingly use contextual information on a daily basis. We can 'situate' ourselves in an environment and process information based on things around us. Studies in Cognitive Psychology [49] provide evidence that contextual cues such as relative size and location play a significant role for object detection in humans.

This is relevant when dealing with velocity estimation because we would tend to predict higher velocities while travelling on highways, as compared to a village road. High-level contextual information has been shown to augment low-level features for object detection tasks [50] achieving better performance. Context has also been shown to play a vital role in 3D scene understanding [51].However, when it comes to velocity estimation, motion blur and ultimately motion flow have been investigated as a cue/feature for velocity estimation, but the role of context is not very well explored. The work by J.Walker et al. [37] predicted motion in static images and deduced that motion prediction is context dependent. Further experiments need to be carried out to cement our understanding of the role of context in velocity estimation.

4

Preliminary results

A review of literature has shown single image frames do contain temporal information which could possibly be utilized to estimate the ego-motion velocity. The first step would in this research work would be to train a convolutional neural network to predict the velocity given single static image frames. If the model is able to learn a mapping from pixel space to velocity, the pertinent question is - what features of the image could contribute towards this task ? To answer that, the transfer learning paradigm can be of use. Transfer learning can be accomplished either by using the CNN as a feature extractor or fine-tuning the previously learned weights on a new possibly related task.

The first section of this chapter discusses and evaluates the data to be used. It is important to understand the input to effectively analyze the predictions of the network. The subsequent sections each analyze the role of motion blur and spatial context in learning a mapping from pixels to velocities.

4.1. Dataset Evaluation

Convolutional neural networks are supervised learning models with a large number of parameters. This means that to efficiently train these models, a large amount of labelled data is required. More specifically, images captured by a moving camera, annotated with the instantaneous velocity at the time of image capture would constitute the dataset for training this deep learning model. Due to the increasing popularity of autonomous driving there are many such datasets available such as the comma.ai dataset, the udacity driving dataset, the KITTI dataset [52] and the Oxford RobotCar dataset[53]. Among these datasets, the KITTI dataset presents itself as the ideal candidate in this scenario as it has a synthetic, generated counterpart name vKITTI [54] which would prove useful. The Oxford dataset is a massive dataset recorded in various scenarios and weather conditions, thus it would prove useful in identifying the edge cases, where the model does not perform as expected.

The KITTI dataset captured images with high resolution colour and grayscale cameras mounted on a Volkswagen station wagon. The cameras recorded 6 hours worth of vehicle motion through varying scenes. A few example frames are depicted in Figure 4.1. High precision GPS/IMU data was recorded alongside the high resolution images. The raw data recorded is divided into five categories depending on the scene, namely Road, Residential, City, Campus and Person. The 'Person' category of the dataset does not involve any ego-motion of the camera, hence it is not considered for evaluation of the deep learning model. The dataset is recorded on 5 different days and always during daytime. The images recorded by the cameras were stored with lossless compression as 8-bit PNG files, with the sky and engine hood cropped out. The information obtained from the GPS/IMU unit such as velocity, acceleration,angular rates were unavailable at times due to outages and in these cases, linear interpolation was used to obtain missing values.

Figure 4.1: Few example frames from KITTI [52]

To obtain a better overview as what kind of motion range is covered by KITTI, the histogram of the ground truth velocity of the dataset is plotted. Also, the mean velocity for each video recorded in each of the four classes(Road, Residential, City, Campus) are visualized to obtain a deeper perspective about motion in each scenario.

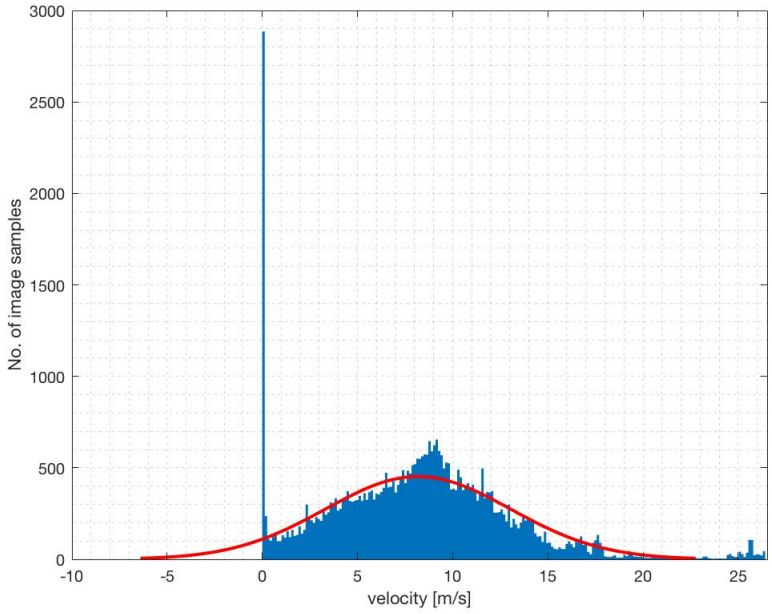

Figure 4.2: Histogram of KITTI velocity ground truth

From the histogram (Figure 4.2), it is evident that the KITTI has a significantly larger amount of images captured at almost zero velocity. However the entire dataset consists of around 43350 images, less than 3000 of which has almost no motion, so this should not pose a problem.

It is evident from Figure 4.3 that the Road and city categories have the highest velocities which is intuitive as these straight, open, highway-like areas, where the vehicle would have been able to travel faster. The campus category has the minimal amount of motion due to the fact that it involves navigating in tight spaces.

The vKITTI dataset is a clone of five videos selected from the original KITTI dataset. The creators of the dataset also automatically generate modified versions of the original sequence(different weather and imaging conditions). A few example frames from vKITTI and their corresponding real world(KITTI) counterpart are depicted in Fig. 4.4 and in this figure it is quite evident that vKITTI is not an exact reproduction. It misses a few objects such as the milestone marker on the road side (second last image) but it captures general aspects of the scene.

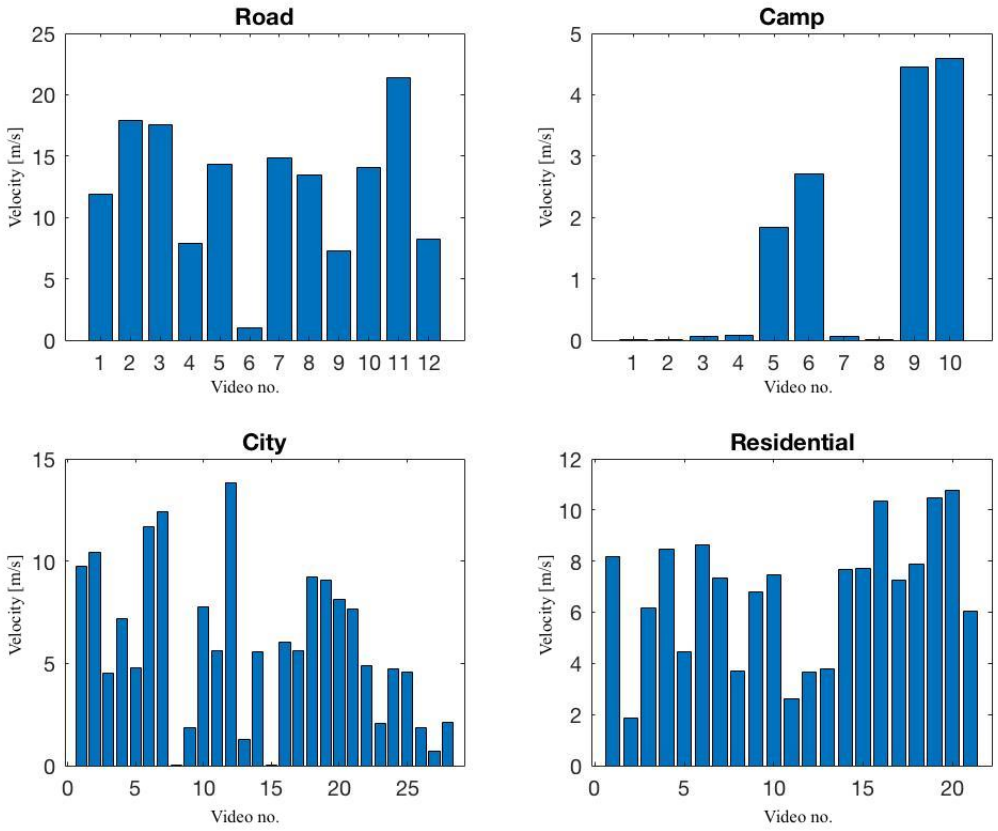

Figure 4.3: Mean velocity distribution for each video recorded per class

Figure 4.4: Few vKITTI image frames alonside KITTI counterparts[54]

The vKITTI dataset does not directly have information of ground truth velocity for the generated scenes, but the extrinsic parameters which capture the transformation from 3D world coordinates to camera coordinates are available. The extrinsic matrix includes the rotation matrix(R) and the world coordinate system origin represented in camera coordinates. Thus, the camera pose can be obtained by the equation : $Pose = -R^T T$. The temporal difference of successive camera poses will in turn result in the ground truth velocities. The computed vKITTI ground truth is compared with the KITTI ground truth for the same frames in Fig. 4.9. There are minor differences between the two ground truths, but the general trend is followed.

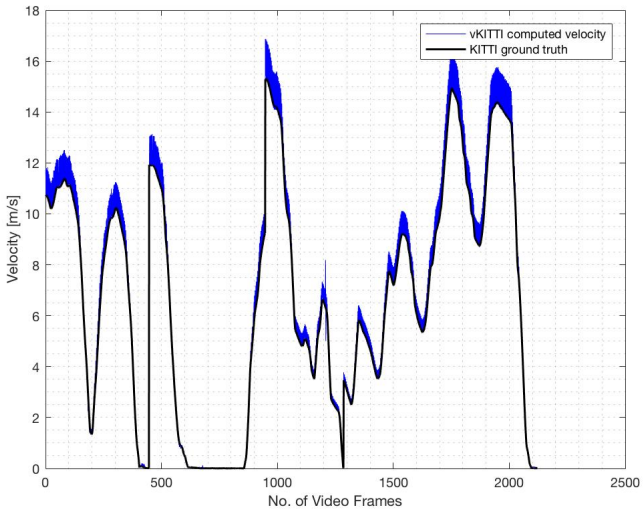

Figure 4.5: Computed vKITTI ground truth and KITTI ground truth

Since vKITTI is a synthetic dataset it is highly probable that it does not include any motion blur. However, as a confirmation step a frame is extracted from KITTI and its corresponding vKITTI frame. The laplacian of the two images are calculated and depicted in Fig. 4.6 and Fig. 4.7. It can be seen that the edges are more clear in vKITTI than with KITTI. Also the median of the top 0.1% pixel value is higher for vKITTI(892.02) than KITTI(791.64). This shows that vKITTI does not have motion blur.

.

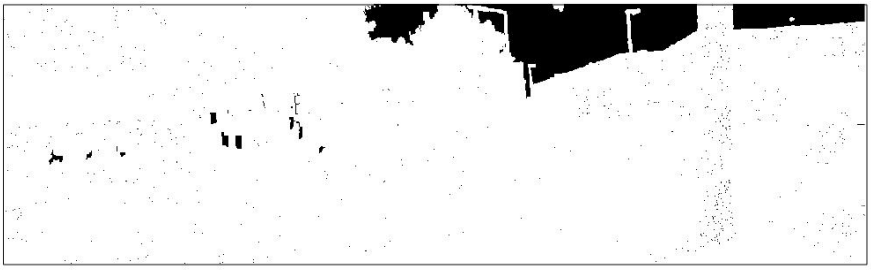

Figure 4.6: Laplacian of a sample KITTI image

Figure 4.7: Laplacian of a sample vKITTI image

To evaluate the role of context in the estimation of camera motion, a dataset with images from a variety of places is required. Scene recognition is a well researched field in computer vision, which means that there is no dearth of datasets. Two of the biggest datasets for scene recognition are the SUN database [55] and the MIT places database [56]. These large datasets share a few common classes but the MIT places dataset is larger,diverse and well documented so it is taken under consideration. There are three macro classes in the MIT places dataset - Indoor, nature and urban which contain several sub-classes of scenes.

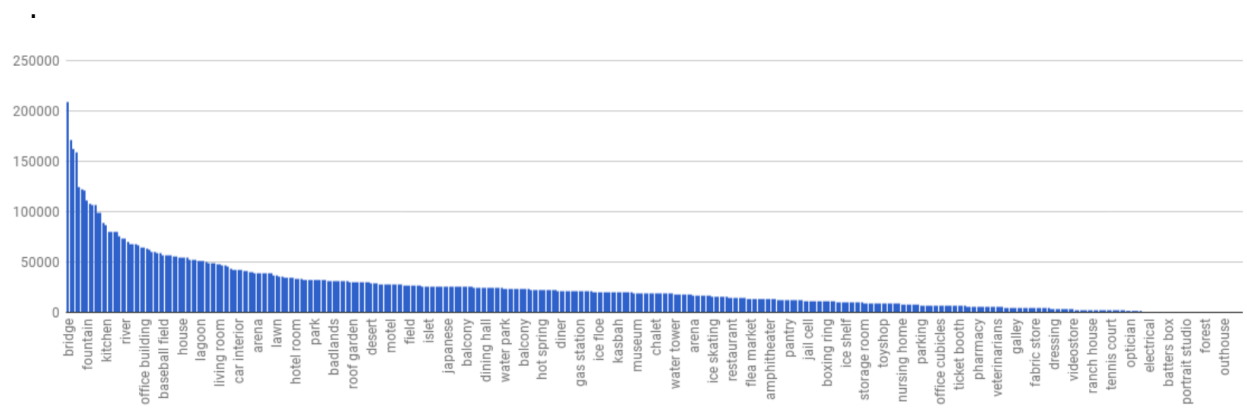

Figure 4.8: Distribution of data - few classes of Places dataset [56]

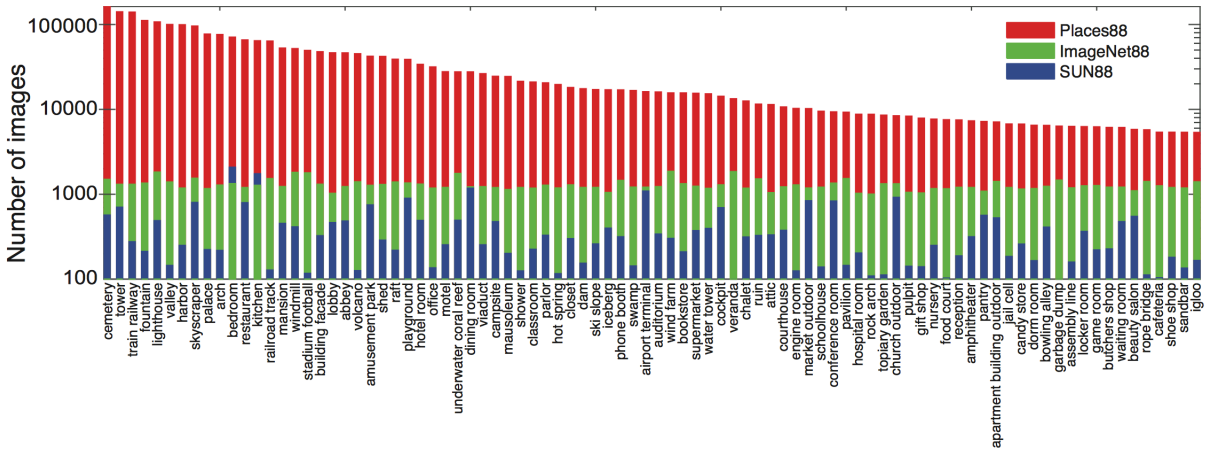

Figure 4.9: Comparison of common 88 classes [56]

The aforementioned datasets aid in establishing the role of motion blur and context in motion estimation. However, these datasets are recorded with cars and thus the motion is planar - cars are incapable of rolling motion along its own axis. Thus it would be interesting to investigate motion estimation with images recorded on board an unmanned aerial vehicle. Such a dataset(drone collected images annotated with velocity) is not available to the best of the authors knowledge, so it must be created. The paparazzi open source autopilot and the motion capture system at Delft University of Technology (Opti-track) are utilized along with the Parrot Bebop drone for the creation of this dataset.

4.2. Experiments

To evaluate the contribution of motion blur and context, the convolutional neural network must first be trained for velocity estimation. An appropriate convolutional neural network architecture must be determined for this task. Given the dynamic nature of the computer vision and deep learning research fields, several convolutional architectures have been developed. These models such as ZFNet [57], GoogleNet [58], VGG [59] and ResNet [60] have all achieved a significant level of performance in the ILSVRC Image classification challenge. Among the above, the VGG16 network architecture has been prevalent, deployed for several tasks such as image style transfer [61], remote sensing image classification [62] and has been showed to learn a generalized set of features which could be attributed to the depth of the network and the fact that it has pooling layer once after every two convolutional layers. The data is relatively less downsampled, allowing it to store more information. The VGG16 architecture is presented in Fig. 4.10

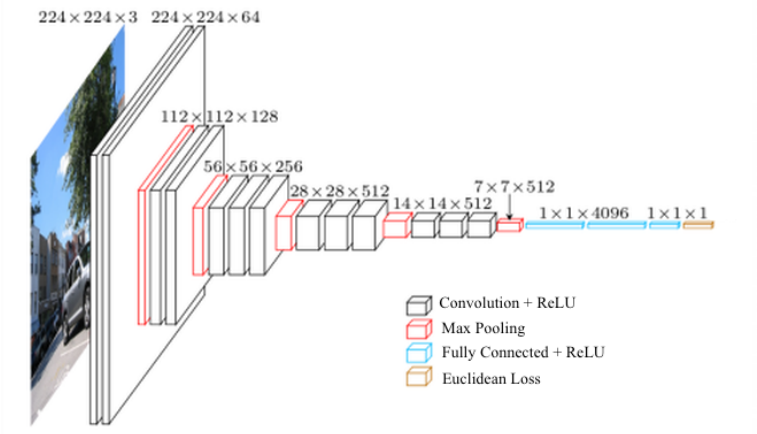

Figure 4.10: VGG16 Convolutional neural network Architecture [63]

The VGG architecture is slightly modified for this scenario. The convolutional layers and the first two fully connected layers are kept intact, the final fully connected layer is modified to output one number - the velocity prediction, rather than the 1000 class scores for imagenet, the last loss layer is changed from Softmax loss to a Euclidean Loss function which is more suited for the regression task.

$$L = \frac{1}{2N}\sum_{i=1}^{N} ||\hat{y}_n - y_n||_2^2 \tag{4.1}$$

The KITTI data is resampled with the nearest neighbour method and then fed into the VGG16 pipeline. The model is trained for 100,000 iterations. The trained model weights are frozen and the model is tested with the test data. The **mean absolute error** for the test set is **3.1585**. A sample of test predictions and targets are shown in the graph below (Fig. 4.11).

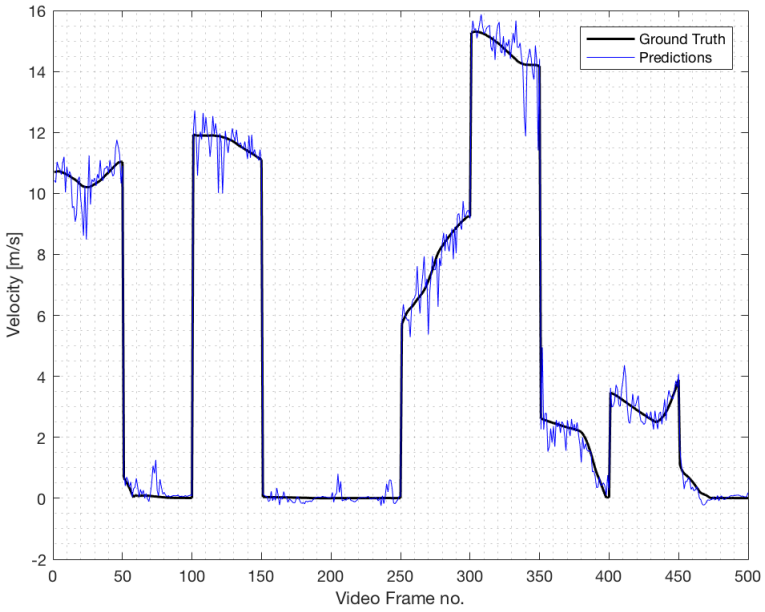

Figure 4.11: Predictions and corresponding ground truth on the test set.

4.2.1. Role of Motion Blur as a feature

To estimate the contribution of motion blur in establishing a mapping from pixel space, to motion space, the trained VGG16 network is used as a feature extractor for a motion blur prediction task. The synthetic dataset, vkitti is used for this task, since it does not have any inherent motion blur. Two datasets are created, one applying various uniform Gaussian blur kernels and another with varying motion blur kernels. A set of 200 random images are sampled from the vKITTI dataset and gaussian convolution of variances ranging from 0.1 to 5 are applied. Along with the original 200, this makes a dataset of 10200 images with increasing uniform blur, annotated with the variance of the kernel used to generate it. The results are shown in Fig. 4.12. While the model struggles to distinguish among the initial kernels,it gets better at prediction with higher variance. This could be due to the fact that at lower variance, the blurred image is similar to the unblurred one.

As with the case of testing with uniform blur, a dataset is created from vkitti for motion blur with blur angle and blur length as the parameters to be estimated. From the results shown in Fig. 4.13, it is evident that the predicted motion blur parameters are very close to the targets.

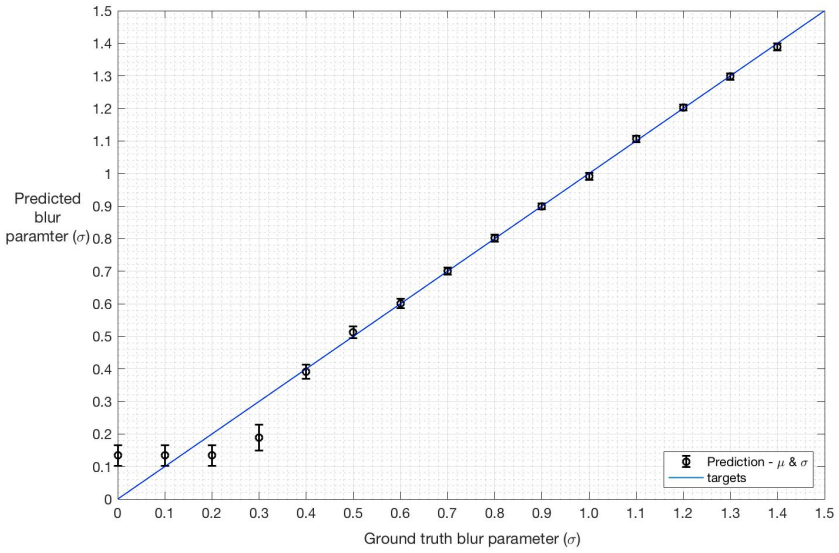

Figure 4.12: Per kernel mean and variance of prediction along with ground truth.

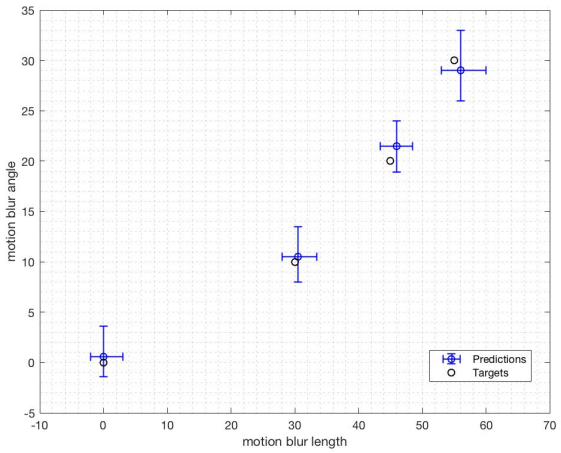

Figure 4.13: Per kernel mean and variance of prediction along with ground truth.

4.2.2. Role of Spatial Context as a feature

To analyze the role of context, a relatively smaller sample of 1.8 million images from the MIT places dataset are supplied as input to the trained convolutional neural network. If the network has learned scene based information, it should predict higher values for highway like scenes in MIT places. However, this does not seem to be the case as can be seen in Fig. 4.14. The predictions from the network have high per class standard deviation. The magenta lines represent predictions on images from indoor classes such as hotel room while the black lines represent outdoor classes like highway road. However, it might be worth noting that some classes have pictures taken indoors and outdoors and there are completely new sets of features that the network has previously not been exposed to. To offer some perspective, the predictions on a few pertinent classes have been depicted in Fig. 4.15. However it can be seen that village has a higher speed prediction as compared to highway.There is also no discernible pattern in the predictions among different classes.

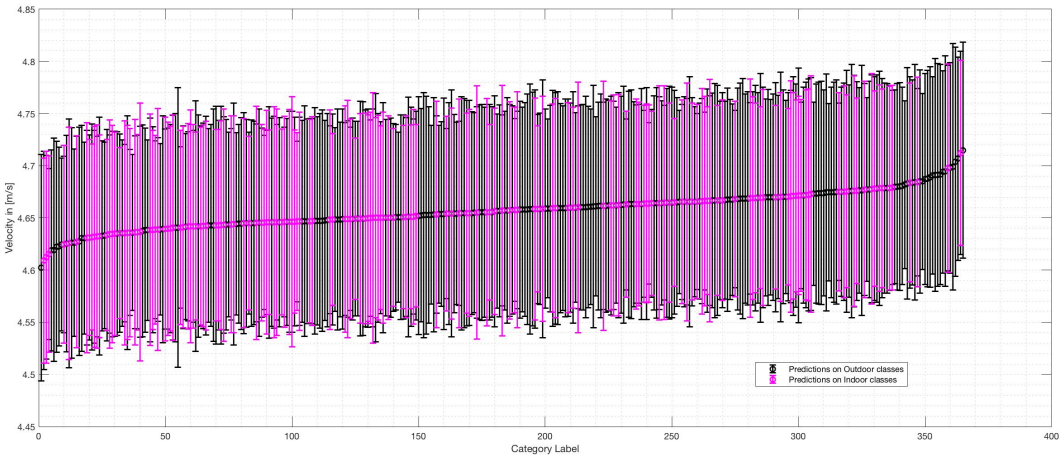

Figure 4.14: Per class mean and standard deviation of Predictions

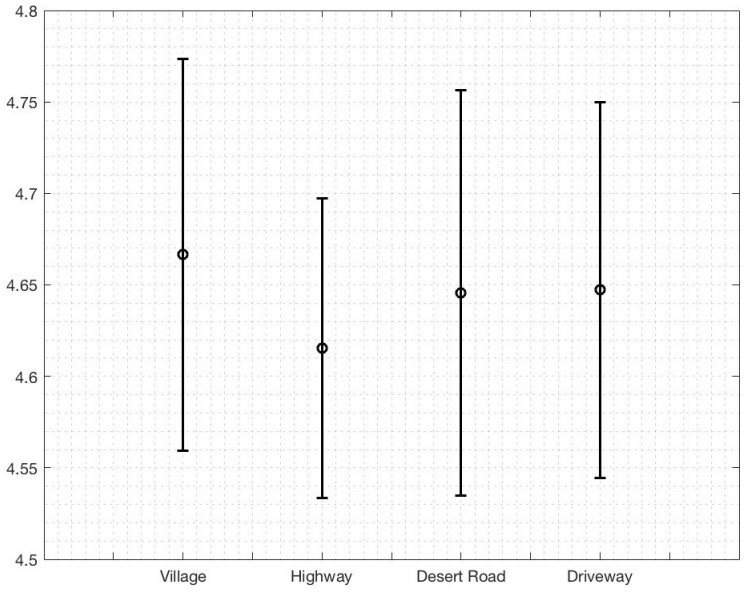

Figure 4.15: Predictions on few relevant classes

5

Conclusion

To summarize the literature survey, the estimation of motion blur is a widely studied problem and in recent times, several deep learning methods were successful in that regard. Single static images have been utilized for motion prediction which implies that single images could potentially be used for velocity estimation. In learning such a mapping between pixels and velocity, the contributing factors are hypothesized to be motion blur and context since these are the cues humans use for speed estimation/motion prediction. The preliminary results show that while motion blur might play a role in this mapping, context might not play any role at all. This can be seen from the fact that the VGG network is able to extract features useful for blur estimation but fails to predict a discernible pattern for the scenes it is exposed to. Further experiments are required to cement our understanding of how the model works.

Bibliography

[1] K. Alexiev and I. Nikolova, *An algorithm for error reducing in imu,* in *Innovations in Intelligent Systems and Applications (INISTA), 2013 IEEE International Symposium on* (IEEE, 2013) pp. 1–6.

[2] M. B. Kjærgaard, H. Blunck, T. Godsk, T. Toftkjær, D. L. Christensen, and K. Grønbæk, *Indoor positioning using gps revisited,* in *International conference on pervasive computing* (Springer, 2010) pp. 38–56.

[3] T. Lemaire, C. Berger, I.-K. Jung, and S. Lacroix, *Vision-based slam: Stereo and monocular approaches,* International Journal of Computer Vision **74**, 343 (2007).

[4] J. Andersh, A. Cherian, B. Mettler, and N. Papanikolopoulos, *A vision based ensemble approach to velocity estimation for miniature rotorcraft,* Autonomous Robots **39**, 123 (2015).

[5] D. Honegger, P. Greisen, L. Meier, P. Tanskanen, and M. Pollefeys, *Real-time velocity estimation based on optical flow and disparity matching,* in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on* (IEEE, 2012) pp. 5177–5182.

[6] V. Grabe, H. H. Bülthoff, and P. R. Giordano, *On-board velocity estimation and closed-loop control of a quadrotor uav based on optical flow,* in *Robotics and Automation (ICRA), 2012 IEEE International Conference on* (IEEE, 2012) pp. 491–497.

[7] Y. LeCun, Y. Bengio, *et al.*, *Convolutional networks for images, speech, and time series,* The handbook of brain theory and neural networks **3361**, 1995 (1995).

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition,* Proceedings of the IEEE **86**, 2278 (1998).

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks,* in *Advances in Neural Information Processing Systems 25*, edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Curran Associates, Inc., 2012) pp. 1097–1105.

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

[11] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, *Flownet: Learning optical flow with convolutional networks,* CoRR **abs/1504.06852** (2015).

[12] D. Gong, J. Yang, L. Liu, Y. Zhang, I. Reid, C. Shen, A. v. d. Hengel, and Q. Shi, *From motion blur to motion flow: a deep learning solution for removing heterogeneous motion blur,* arXiv preprint arXiv:1612.02583 (2016).

[13] J. Sun, W. Cao, Z. Xu, and J. Ponce, *Learning a convolutional neural network for non-uniform motion blur removal,* in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[14] H.-Y. Lin and K.-J. Li, *Motion blur removal and its application to vehicle speed detection,* in *Image Processing, 2004. ICIP'04. 2004 International Conference on*, Vol. 5 (IEEE, 2004) pp. 3407–3410.

[15] R. Mebarki, J. Cacace, and V. Lippiello, *Velocity estimation of an uav using visual and imu data in a gps-denied environment, 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics*, (2013).

[16] K. McGuire, G. de Croon, C. De Wagter, K. Tuyls, and H. Kappen, *Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone,* IEEE Robotics and Automation Letters , 1070 (2017).

[17] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, *Vision-based state estimation for autonomous rotorcraft mavs in complex environments,* in *Robotics and Automation (ICRA), IEEE International Conference on* (IEEE, 2013) pp. 1758–1764.

[18] S. Weiss, D. Scaramuzza, and R. Siegwart, *Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments,* Journal of Field Robotics **28**, 854 (2011).

[19] S. L. Pintea, J. C. van Gemert, and A. W. Smeulders, *Déja vu,* in *European Conference on Computer Vision* (Springer, 2014) pp. 172–187.

[20] H.-Y. Lin, K.-J. Li, and C.-H. Chang, *Vehicle speed detection from a single motion blurred image,* Image and Vision Computing **26**, 1327 (2008).

[21] S. S. L. Oskouei, H. Golestani, M. Kachuee, M. Hashemi, H. Mohammadzade, and S. Ghiasi, *Gpu-based acceleration of deep convolutional neural networks on mobile platforms,* CoRR **abs/1511.07376** (2015).

[22] H. Bal, D. Epema, C. de Laat, R. van Nieuwpoort, J. Romein, F. Seinstra, C. Snoek, and H. Wijshoff, *A medium-scale distributed system for computer science research: Infrastructure for the long term,* Computer **49**, 54 (2016).

[23] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, *Caffe: Convolutional architecture for fast feature embedding,* CoRR **abs/1408.5093** (2014), arXiv:1408.5093 .

[24] W. S. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity,* The bulletin of mathematical biophysics **5**, 115 (1943).

[25] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning* (MIT press, 2012).

[26] Veličković, Petar, *Deep learning for complete beginners: convolutional neural networks with keras,* (2017).

[27] Karpathy, Andre, *Cs231n convolutional neural networks for visual recognition,* .

[28] Gulli, Antonio, *Convolutional neural networks with reinforcement learning,* .

[29] Bullinaria, John, *Recurrent neural networks, neural computation : Lecture 12,* .

[30] D. Eigen, C. Puhrsch, and R. Fergus, *Depth map prediction from a single image using a multi-scale deep network,* CoRR **abs/1406.2283** (2014), arXiv:1406.2283 .

[31] H. N. Ramakrishnan, *Detection and Estimation of Image Blur*, Master's thesis, Missouri University of Science and Technology (2010).

[32] A. Savakis and R. L. Easton, *Blur identification based on higher order spectral nulls, Proceedings of SPIE - The International Society for Optical Engineering,* **2302** (1994).

[33] T.-L. Pao and M.-D. Kuo, *Estimation of the point spread function of a motion-blurred object from autocorrelation,* , 1226 (1995).

[34] Y. Yitzhaky and N. Kopeika, *Identification of blur parameters from motion blurred images,* Graphical Models and Image Processing **59**, 310 (1997).

[35] R. Dash, P. Kumar Sa, and B. Majhi, *Rbfn based motion blur parameter estimation, 2012 International Conference on Advances in Computing and Communications*, (2009).

[36] H. W. Ho, G. C. de Croon, and Q. Chu, *Distance and velocity estimation using optical flow from a monocular camera,* International Journal of Micro Air Vehicles **9**, 198 (2017).

[37] J. Walker, A. Gupta, and M. Hebert, *Dense optical flow prediction from a static image,* in *Computer Vision (ICCV), 2015 IEEE International Conference on* (IEEE, 2015) pp. 2443–2451.

[38] B. K. Horn and B. G. Schunck, *Determining optical flow,* Artificial intelligence **17**, 185 (1981).

[39] B. D. Lucas and T. Kanade, *An iterative image registration technique with an application to stereo vision,* in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1981) pp. 674–679.

[40] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, *Deepflow: Large displacement optical flow with deep matching,* in *Computer Vision (ICCV), 2013 IEEE International Conference on* (IEEE, 2013) pp. 1385–1392.

[41] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, *Epicflow: Edge-preserving interpolation of correspondences for optical flow,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015) pp. 1164–1172.

[42] M. J. Black, Y. Yacoob, A. D. Jepson, and D. J. Fleet, *Learning parameterized models of image motion,* in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on* (IEEE, 1997) pp. 561–567.

[43] J. L. D. Sun, S. Roth and M. J. Black., *Learning optical flow,* in *European Conference on Computer Vision* (2008).

[44] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, *Convolutional learning of spatio-temporal features,* in *European conference on computer vision* (Springer, 2010) pp. 140–153.

[45] K. Konda and R. Memisevic, *Unsupervised learning of depth and motion,* arXiv preprint arXiv:1312.3429 (2013).

[46] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, *Flownet 2.0: Evolution of optical flow estimation with deep networks,* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2 (2017).

[47] K. Purohit, A. Shah, and A. N. Rajagopalan, *Bringing Alive Blurred Moments!* ArXiv e-prints (2018), arXiv:1804.02913 [cs.CV] .

[48] M. Jin, G. Meishvili, and P. Favaro, *Learning to Extract a Video Sequence from a Single Motion-Blurred Image,* ArXiv e-prints (2018), arXiv:1804.04065 [cs.CV] .

[49] I. Biederman, R. J. Mezzanotte, and J. C. Rabinowitz, *Scene perception: Detecting and judging objects undergoing relational violations,* Cognitive psychology **14**, 143 (1982).

[50] R. Girshick, J. Donahue, T. Darrell, and J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014) pp. 580–587.

[51] Y. Zhang, M. Bai, P. Kohli, S. Izadi, and J. Xiao, *Deepcontext: Context-encoding neural pathways for 3d holistic scene understanding,* arXiv preprint arXiv:1603.04922 (2016).

[52] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, *Vision meets robotics: The kitti dataset,* The International Journal of Robotics Research **32**, 1231 (2013).

[53] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, *1 year, 1000 km: The oxford robotcar dataset,* The International Journal of Robotics Research **36**, 3 (2017).

[54] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, *Virtual worlds as proxy for multi-object tracking analysis,* in *CVPR* (2016).

[55] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, *Sun database: Large-scale scene recognition from abbey to zoo,* in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on* (IEEE, 2010) pp. 3485–3492.

[56] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, *Places: A 10 million image database for scene recognition,* IEEE Transactions on Pattern Analysis and Machine Intelligence (2017).

[57] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks,* CoRR **abs/1311.2901** (2013), arXiv:1311.2901 .

[58] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *et al.*, *Going deeper with convolutions,* (Cvpr, 2015).

[59] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition,* arXiv preprint arXiv:1409.1556 (2014).

[60] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition,* CoRR **abs/1512.03385** (2015), arXiv:1512.03385 .

[61] L. A. Gatys, A. S. Ecker, and M. Bethge, *Image style transfer using convolutional neural networks,* in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on* (IEEE, 2016) pp. 2414–2423.

[62] G. Cheng, Z. Li, X. Yao, L. Guo, and Z. Wei, *Remote sensing image scene classification using bag of convolutional features,* IEEE Geoscience and Remote Sensing Letters **14**, 1735 (2017).

[63] Pawit Kocakarn, *Machine learning with python : Image classifier using vgg16 model - part 1: Theory,* .