

# Cyclist Collision Detection in Urban Environments

## — Background Report —

Peter Hedley  
ph817@doc.ic.ac.uk

Supervisor: Dr. Stefan Leutenegger  
Course: CO541, Imperial College London

20<sup>th</sup> July, 2018

### **Abstract**

This report investigates the use of existing state of the art image detection, segmentation and SLAM algorithms for use by cyclists in urban environments. The challenges facing cyclists in a daily commute include monitoring pedestrian movement, vehicle movement and obeying the rules of the road. The aim of the project is to provide a detection system to enable safer cycling on dangerous city-roads.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Background, Computer Vision</b>	<b>4</b>
2.1	Deep Learning Preliminaries . . . . .	4
2.2	Convolutional Networks . . . . .	4
2.2.1	Convolutional Layers . . . . .	5
2.2.2	Pooling Layers . . . . .	5
2.3	Image Detection . . . . .	5
2.3.1	R-CNN . . . . .	6
2.3.2	Fast-RCNN . . . . .	6
2.3.3	Faster R-CNN . . . . .	7
2.4	Image Segmentation . . . . .	8
2.5	Instance Segmentation . . . . .	9
2.5.1	Mask R-CNN . . . . .	10
<b>3</b>	<b>Background, SLAM</b>	<b>10</b>
3.1	Sensors . . . . .	11
3.2	World and Camera Frame . . . . .	11
3.3	Bicycle States . . . . .	12
3.4	The SLAM Problem . . . . .	12
3.4.1	Camera Re-projection Error . . . . .	13
3.4.2	IMU Kinematics . . . . .	13
3.4.3	IMU Kinematics Linearised . . . . .	14
3.4.4	Keypoint Detection . . . . .	14
<b>4</b>	<b>Background, Object Tracking</b>	<b>14</b>
4.1	CamShift . . . . .	15
4.2	Object Tracking Summary . . . . .	15
<b>5</b>	<b>ROS</b>	<b>16</b>
<b>6</b>	<b>Frame-to-Frame Tracking</b>	<b>16</b>
6.1	Intersection Over Union . . . . .	17
6.2	BRISK - Selector . . . . .	17
6.3	Class Matching . . . . .	18
6.4	Kalman Filter ROI Matching . . . . .	18
6.4.1	Kalman Predictive Step . . . . .	18
6.4.2	Kalman Update Step . . . . .	19
6.4.3	Kalman Equations . . . . .	19
6.4.4	Kalman Matching ROIS . . . . .	19
6.5	Combining Matching Methods . . . . .	20
6.6	ROI Lives . . . . .	20
6.7	Combining Match Predictions . . . . .	20
<b>7</b>	<b>Ethics Checklist</b>	<b>20</b>

## 1 Introduction

There were c. 18,500 cycling accidents and 3,500 deaths on British roads in 2016 [1]. Of these accidents 80% occur during the day and driver/rider error accounts for 71% of the total collisions [2]. The most common place for a cycling incident is at a road junction with 75% of collisions occurring there [3]. With cycling becoming a more accepted form of transport, the need to protect and modify bikes with a comprehensive safety system is important.

This report will examine state of the art technologies that are used for autonomous vehicles to investigate their applicability to a cycling environment. The three main topics involve image detection/segmentation, SLAM algorithms and collision/path prediction.

## 2 Background, Computer Vision

### 2.1 Deep Learning Preliminaries

Computer vision and its applications has been revolutionised recently by Deep Learning frameworks. These have led to super-human accuracy in image recognition, for example Inception v3 [4], release in 2016, reached a 3.46% error rate on the ImageNet [5] dataset. This is known as an image classification task; the photo contains one object in the dataset and that object generally occupies the majority of the frame. Similar networks are used with additional features for image detection and segmentation which are integral to detecting moving objects such as pedestrians and vehicles in an urban environment.

The current state-of-the-art methods in computer vision use a form of neural network. These networks are characterised by a series of non-linear nodes that are organised in layers. At each layer the network learns progressively abstract and higher-level features of the image to be detected until the final (usually dense layer) which determines the predicted class. Therefore the 'depth' of a neural network often refers to the number of layers and hence the inherent complexity of features that a network can learn. This is, however, only a basic representation of the problem and some clever and shallower architectures are able to outperform deeper networks. It is important to note that the depth and width, more specifically the number of parameters in the model generally reduces speed of prediction and increases the memory required to store/load the model into RAM. This is of little consequence in most applications, however for the limited computational power that can be carried on a bicycle it will be of concern in this report.

### 2.2 Convolutional Networks

A CNN generally contains convolutional, pooling, fully connected and normalisation layers. This form of network became particularly popular in computer vision after Geoffrey Hinton's 2012 paper [6], with an ImageNet classification error of c.16%, significantly lower than the state-of-the-art at the time which was 26%. Convolutional networks use the assumption that each input is an image and therefore certain properties can be encoded into the architecture of the network. This allows a significant reduction in the number of tunable network parameters.

A regular neural network, as mentioned earlier Section 2.1, contains a number of fully connected layers i.e. each neuron in a layer is connected to every neuron in the previous and next layers respectively. However convolutional neural networks are arranged in three dimensions: width, height, depth (Figure 1). In this architecture the neurons in a layer are only connected to some of the layer before, unlike in the fully connected case, this significantly reduces the number of parameters [7].

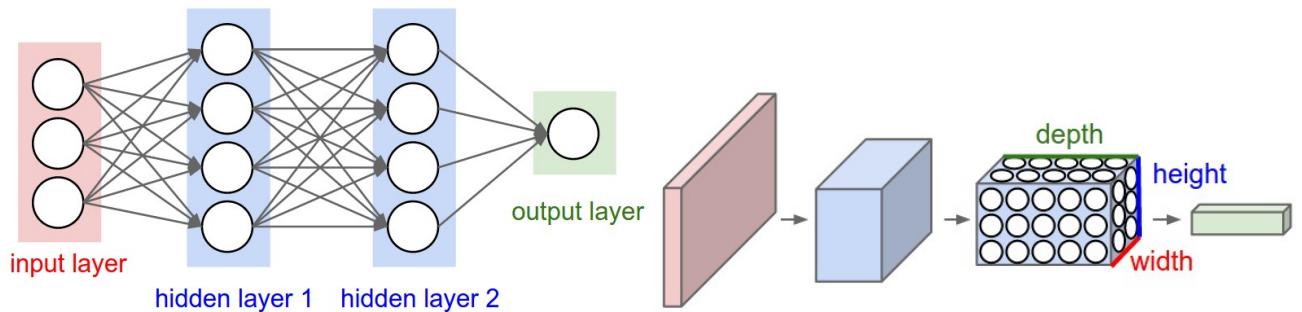


Figure 1: Regular Neural Network (Left) [7] , CNN (right) [7].

### 2.2.1 Convolutional Layers

A convolutional layer learns a set of filters, each filter is compact in size, say  $(7 \times 7 \times 3)$ . This filter slides (convolves) over the image, then a dot product is taken between the filter values and the inputs to that filter (Figure 2). The filter slides across with a respective stride, which is a hyperparameter, a stride of one equates to moving one pixel across at a time. A convolutional layer is therefore able to pick up features such as a corner, or higher level features such as the texture of fur. This process creates a 2D activation map containing the response of the filter at each position on the input volume. A number of filters are passed over and a number of activation maps are created, relating to the layer depth which is a tuneable hyperparameter.

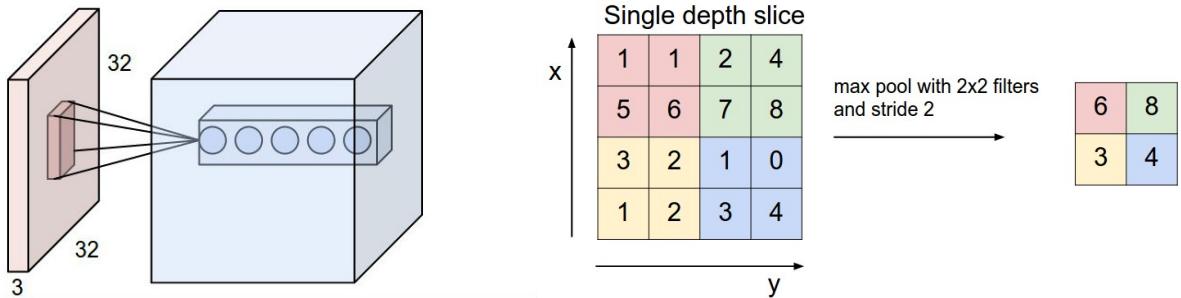


Figure 2: CNN Filter (left) [8], Max Pooling [9]

### 2.2.2 Pooling Layers

Pooling layers are used to reduce the number of parameters in a network. This is a much debated topic, Geoffrey Hinton's paper [10] discusses the problems of convolutional networks in his 2017 paper and especially focuses on the disadvantages of pooling layers. It is however a current, although fading, method of parameter reduction. For the purposes of optimising network architecture for smaller hardware such as is portable on a bike this is still a recognised and useful method.

A pooling layer such as that depicted in (Figure 2) passes a kernel over the input (much like in a convolutional layer) and at each spacial location it calculates a metric. For example, max pooling takes the largest value contained within the kernel grid at each consecutive spacial location. This process reduces the input size with respect to the kernel size and the implemented stride.

## 2.3 Image Detection

Image detection is an area of research that uses the CNN networks described in Section 2.2. This is, however, a more difficult task as there can be multiple objects/classes of interest within one image. A naive approach to this problem would be to slide a window across an image and classify at each location. When the network classifies, say a cat at a particular window location it is likely that a cat is located in that area.

Although the aforementioned approach is valid it would be very slow, not only would it be needed to slide the window over the whole image, the size of the window would also need to be varied. The network cannot know the area of the photo that the object occupies and therefore what size to make the window. There are a number of differing approaches to this problem R-CNN [11] and other later iterations vs YOLO (you only look once) [12]. The R-CNN approach is very similar to the vanilla, naive solution, discussed above (commonly known as exhaustive search) and is used by later techniques for instance segmentation.

### 2.3.1 R-CNN

R-CNN, proposed by Ross Girshick et al. [11] in 2014, incorporates a region proposal method. This process produces 2000 proposals which are then passed through a large CNN, and each region is classified using class-specific linear SVMs. This process equates to a 53.7% mean average precision (mAP) compared to 35.1% by state-of-the-art methods at the time. In addition to this performance gain the system doesn't rely on far more complicated ensemble methods.

The region proposal method used is called selective search [13], this uses a fast segmentation algorithm by Felzenszwalb [14] followed by a greedy algorithm to iteratively group regions together using similarity metrics. The four metrics proposed in the paper are  $s_{colour}$ ,  $s_{texture}$ ,  $s_{size}$ ,  $s_{fill}$  the aim is to use a number of metrics to pick up the differences between regions during the growth period and also limit one region's growth at the expense of others (hence the size metric).

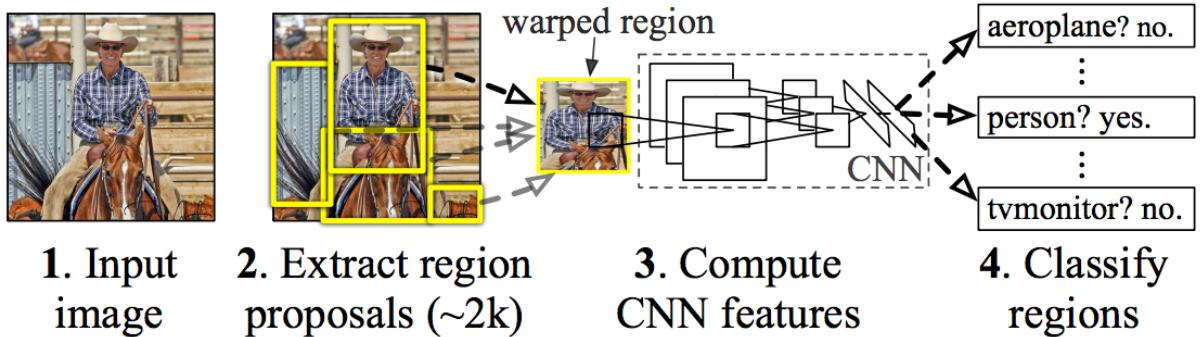


Figure 3: R-CNN Region Proposal & Architecture [11]

The results of the region proposal (Figure 3) are a series of windows that are likely to contain an object. These are then simply warped to  $227 \times 227$  pixels to fit the input dimension's of the CNN network that follows, containing five convolutional layers and two fully connected layers. The output of the CNN is then fed as a 4096 dimensional feature vector to a set of trained SVMs, one for each class, that are used to score the likelihood for that class. This approach is slow at 13s/image on a GPU and 53s/image on a CPU, the separation of a CNN and SVM is peculiar and adds to this timescale. A later iteration of this method Fast-RCNN is discussed in Section 2.3.2.

### 2.3.2 Fast-RCNN

R-CNN performs a CNN forward pass for each region proposal and does not share computation between regions [15]. Spacial pyramid pooling networks (SPPnet) [16] are designed to mitigate this problem by creating a feature map (which is always the same shape) from an input image and then classifying each object proposal from a vector extracted from the shared feature map. This paper still uses a multiple-stage classification pipeline but is a marked improvement on R-CNN with regards to speed.

Fast-RCNN uses the SPPnet concept but also incorporates a single-stage pipeline. The process inputs an image and multiple RoIs into a fully convolutional network (Figure 4). Each ROI is then pooled into a fixed-size feature map as in SPPnet and then mapped to a feature vector by fully connected layers. As can be seen in Figure 4 the network then branches into 2 separate outputs, one 4-point regression output of the bounding box location and a K-size classification output, where K is the number of object classes. The model therefore must contain a multi-task loss function that accounts for both bounding box and softmax loss (Equation 1).

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v) \quad (1)$$

in which  $L_{cls}(p, u) = -\log(p_u)$  is the log-loss for true class  $u$ , and  $L_{loc}$  is the bounding box regression loss defined as:

$$L_{loc} = \sum_{i \in \{x, y, w, h\}} smooth_{L_1}(t_i^u - v_i), \quad (2)$$

in which

$$smooth_{L_1} = \begin{cases} 0.5x^2 & \text{if } x < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \quad (3)$$

is a  $L_1$  loss. This architecture allows for fast training with the forward pass of the network and therefore processes images  $146\times$  faster than R-CNN due to sharing computation between layers and the improved pipeline design.

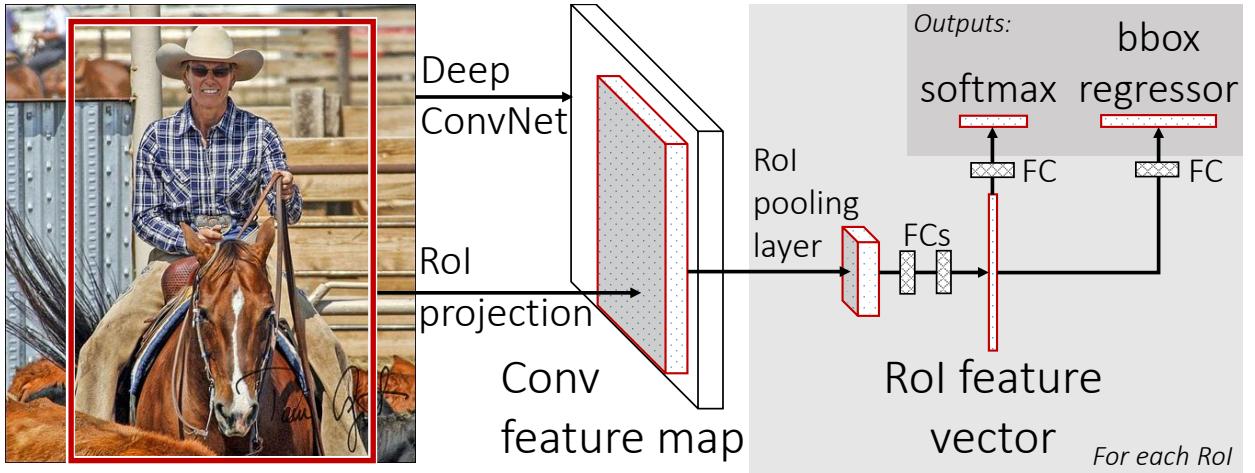


Figure 4: Fast R-CNN Architecture [15]

Figure 4 depicts the network architecture, this incorporates an RoI pooling layer which maps the convolutional feature map from an RoI (created by selective search) to a fixed extent feature map of a set  $H \times W$ , where  $H$  and  $W$  are layer hyperparameters. Each RoI is a rectangular window defined by a tuple of four elements  $(r, c, h, w)$ , where  $(r, c)$  specify the top left corner and  $(h, w)$  the width. RoI max pooling then divides the RoI window into a grid and then max-pools each grid into the corresponding output cell. The size of the grid is proportionate to the RoI and so the output shape is constant regardless of RoI input size.

### 2.3.3 Faster R-CNN

Although the previous method [15] removed the need for a multi-stage classification pipeline (CNN and SVM), it still relies on the selective search [13] method for region proposal. This part is now the computational bottleneck that slows down the process. Faster R-CNN proposed by Ren et al. [17] seeks to alleviate this issue by combining region proposal into the CNN. The paper suggests the use of a region proposal network (RPN) to produce the RoIs used later in the network.

The region proposal network (RPN) outputs a set of rectangular object proposals, each of these proposals has an objectiveness score i.e. how likely the network thinks the proposal contains an object. This part of the network comes after a number of convolutional layers (in this paper 5 or 13 such layers). The region proposals are then created off the feature map from these layers by sliding a smaller network of size  $n \times n$  over the convolutional feature map (Figure 5). At each sliding window location a number of region proposals are predicted simultaneously. These consist of 3 aspect ratios and 3 scales, therefore 9 total region predictions or anchors at each location. The network is then followed

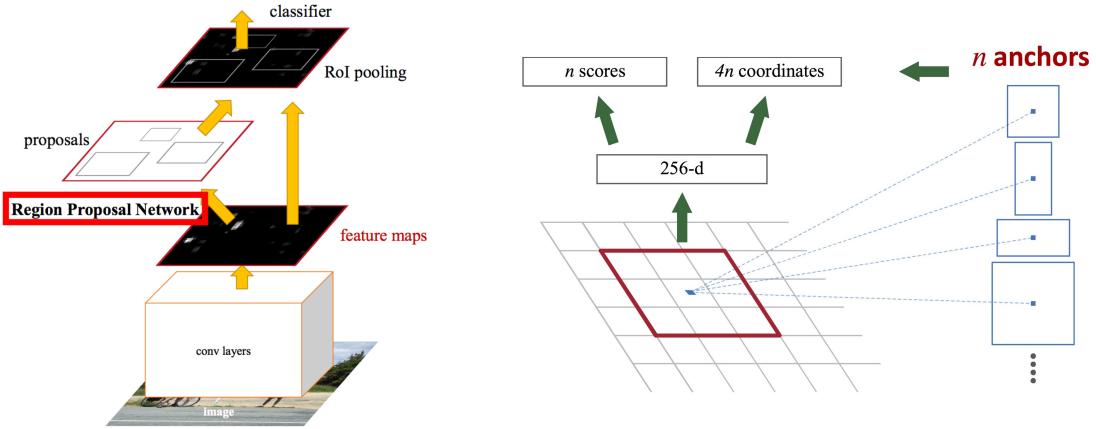


Figure 5: Fast R-CNN Architecture [15]

by two sibling  $1 \times 1$  convolutional layers, one for  $reg$  (region proposal) and one for  $cls$  (probability of object or not). The  $reg$  layer outputs  $4k$  results, which represents the encoding coordinates of the  $k$  boxes and the  $cls$  later scores the probability that there is an object or no object in each proposal, therefore  $cls$  has  $2k$  outputs. This results in a convolutional layer output of depth 9.

The loss for the network is calculated from two metrics, the output of the  $reg$  layer is a binary, i.e. positive example, or not. A positive example is defined by how much the box overlaps with the ground truth box of the object and so an IoU (Intersection-over-union) overlap  $> 0.7$  is considered a positive example and an IoU  $< 0.3$  is considered a negative label. For the first positive case, if no positive examples are found, the boxes with highest IoU are labelled positive. With this it follows that:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (4)$$

Where  $i$  is the anchor index,  $p_i$  is the predicted probability of anchor  $i$  containing an object. The ground truth label  $p_i$  is denoted as a 1 for a positive anchor and 0 for a negative therefore  $L_{reg}$  is only activated for a positive example,  $t_i$  and  $t_i^*$  represent the 4 coordinates of the bounding boxes, and  $L_{cls}$  is the classification loss. The two terms are normalised ( $N_{cls}$  and  $N_{reg}$ ) and balanced by a weighting parameter  $\lambda$ . The final term to discuss, the bounding box regression is calculated, simply, as follows:

$$\begin{aligned} t_x &= (x - x_a)/w_a, & t_y &= (y - y_a)/h_a, \\ t_w &= \log(w/w_a), & t_h &= \log(h/h_a), \\ t_x^* &= (x^* - x_a)/w_a, & t_y^* &= (y^* - y_a)/h_a \\ t_w^* &= \log(w^*/w_a), & t_h^* &= \log(h^*/h_a) \end{aligned} \quad (5)$$

where  $x$ ,  $y$ ,  $w$ , and  $h$  are the box centre coordinates plus width and height. Predicted box, and anchored box are denoted by  $x$ ,  $x_a$ ,  $x^*$  (likewise for  $y$ ,  $w$ ,  $h$ ).

Finally after the regions have been proposed, c.6000 per image, some regions overlap with others, and so, to reduce redundancy non-maximum suppression is used on their  $cls$  score with an IoU threshold of 0.7 to leave around 2000 proposals per image. Non-maximum suppression is a technique that sets all neighbouring areas to zero around a local maximum and hence only keeps the maximum value and suppresses the others.

## 2.4 Image Segmentation

Unlike the image detection problem described in Section 2.3, this task aims to segment an image on a pixel-by-pixel basis into respective classes (Figure 6). A conventional classification network takes an image and outputs a prediction of which class the image is i.e. a vector of probabilities or a

‘score’. This process uses fully-connected layers to produce the final output vector which throws away important spacial information used in a segmentation task. Therefore Long et al. [18] propose converting these final layers to convolutional layers, maintaining the spacial information and creating a heatmap output (Figure 6).



Figure 6: Image Segmentation [19], Convolutional Heat Map [18]

The heatmaps produced from the network then need to be upsampled from their respective dimensions to that of the output image. This process is essentially a reverse or deconvolution. Upsampling, therefore, with factor  $f$  is convolution with a fractional input stride of  $1/f$ . Essentially each filter is placed on the output image and then multiplied by the input pixel. Hence the image increases in size depending on stride and kernel size as one pixel value is interpolated over a grid. The final problem therefore is how to choose the filter values, this can be done in a number of ways but the main accepted approach is to simply learn them as part of the tuning process.

An important concept is that of patchwise training which feeds the network a number of patches from an input image (small patches surrounding objects of interest) instead of an entire input image. This both helps to balance the classes, ensures the input has enough variance, and is a correct representation of the input set. However, this paper argues that this can be done from a fully convolutional training regime with incorporating a DropConnect-like mask [20] between the output and the loss. A DropConnect mask is similar to the proposed regularisation technique of Dropout [21] except that it randomly sets the network weights to 0 rather than the activation functions. This, on the output layer (connected to the network loss) is found to have the same effect as patch-wise sampling in Long et al.’s paper.

Although conventional proven classification architectures performed to state-of-the-art when modified for segmentation (56.0 mean IU), Long at al. [18] go on to design a bespoke network that achieves 62.7 mean IU. Mean pixel intersection over union (Mean IU) is a standard performance metric where the mean is taken over all classes, including the background. The paper describes a network architecture that uses skips [22]. Skips allow predictions from lower-level coarser layers to interact with finer, latter layer predictions. This lets the model make local predictions (finer) within the context of the global structure (coarser). This addition made for a marked increase in performance by simply up-sampling earlier layers and summing with latter layer outputs for a final image prediction.

## 2.5 Instance Segmentation

Instance segmentation is a problem that combines research in Sections 2.3 & 2.4, it aims to not only perform pixel-wise segmentation of classes but also determine instances of objects within an image. As discussed in Section 2.5.1 there is a simple method of combining these networks into one simple classification pipeline.

### 2.5.1 Mask R-CNN

Mask R-CNN was proposed by Facebook AI Research (FAIR) namely Kaiming He et al. in March 2017. It uses the Faster R-CNN network described in Section 2.3.3 which builds from advances in Sections 2.3.1 & 2.3.2. The method works by simply adding a branch from the existing network for predicting segmentation masks as in Section 2.4 (Figure 7). The branch is a small FCN that is applied to each ROI for segmentation. There are a number of slight modifications to each approach to produce the final Mask R-CNN that are documented below.

RoIs are produced from the RPN as in Faster R-CNN [17] then an  $m \times m$  mask is predicted from

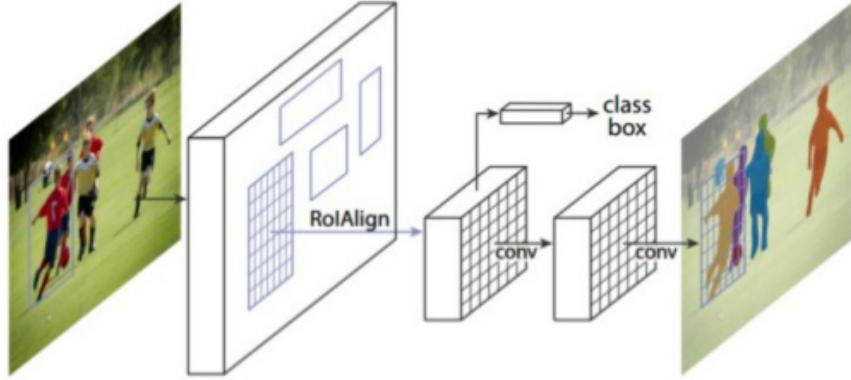


Figure 7: Mask R-CNN [23]

each ROI without the use of a fully-connected (fc) layer and therefore keeps spacial information as in Section 2.4. It was found during experimentation that it is preferable to decouple the class classification and the mask production of the network. This stops multiple classes disrupting eachother during mask production, hence a mask is produced for each class. The class of each respective mask is then determined from the class output branch and that binary mask is selected.

The loss function for the network must now incorporate the three different branches; mask, class prediction, bounding-box prediction (Equation 6).

$$L = L_{cls} + L_{box} + L_{mask} \quad (6)$$

Where  $L_{cls}$  &  $L_{box}$  are identical to Section 2.3.2. The mask branch loss is determined by a per-pixel sigmoid between the ground-truth class ( $k$ ) and the corresponding  $k$ th binary mask (excluding other classes) and is calculated with an average binary cross-entropy loss.

RoI pool as discussed in Section 2.3.2 is modified in this approach. Instead of subdividing an ROI into discrete regions e.g. a grid of  $4 \times 5$  cells. An ROI does not always divide equally into these grid cells and so rounding of grid sizes is needed, this quantisation of sub-windows can cause misalignments of the ROI and extracted features. The new RoIAlign layer introduced in this paper mitigates this problem by not rounding grid-cells. Mask R-CNN uses bi-linear interpolation (linear interpolation with 2d) to compute the exact values of input features at four regularly samples locations in each ROI window.

## 3 Background, SLAM

In order to detect collisions between the bike and obstacles the software must have an understanding of the global environment. The detection described in Section 2 allows mapping, and tracking of dynamic objects in the frame, but distance, path prediction and eventually collision prediction rely strongly on an accurate understanding of 2.5-3D space.

### 3.1 Sensors

The sensors used in this project are an RGBD camera and an IMU. These are both available in the Intel Realsense Camera (ZR300, Figure 8). This camera has a USB cable that both provides the power and sends camera information. An extension cable allows for a laptop to be attached to the camera, from a backpack whilst cycling. Potential problems with this setup are due to juddering of the bike during operation; this could cause tricky footage to analyse and can data loss when the hard-drive is writing to disk. The later issue can easily be solved by writing to an SSD while cycling.



Figure 8: Intel Realsense Camera (ZR300)

### 3.2 World and Camera Frame

In order to fully understand the bike's location, the environment can be considered from two frames of reference, the world and camera frame.

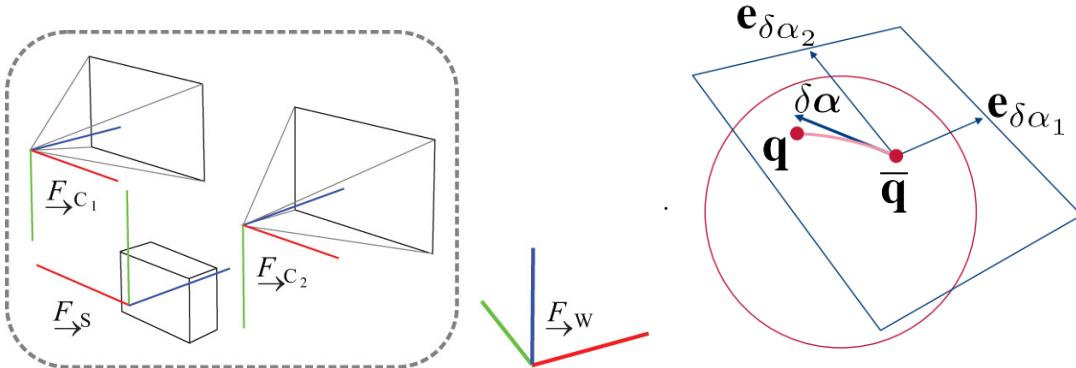


Figure 9: World vs Camera Frame [24]

To convert between world ( $\overset{F}{\rightarrow}_W$ ), camera frame ( $\overset{F}{\rightarrow}_{C_i}$ ), and IMU frame ( $\overset{F}{\rightarrow}_S$ ), an appropriate method of describing the orientation, and location of an object in 3D space is needed. A Hamiltonian Quaternion is capable of correctly mapping an object in 3D space, it does this by use of a 4D representation of one real part, and 3 imaginary;

$$q = q_w + q_x i + q_y j + q_z k, \quad (7)$$

$$i^2 = j^2 = k^2 = ijk = -1$$

The Vectors expressed within a frame are denoted as  $\mathbf{p}_A$  or  $\mathbf{p}_A^{BC}$  where B and C are start and end points. A transformation between frames is represented as  $\mathbf{T}_{BA}$  where B and A represent reference frames, so  $\mathbf{T}_{WC}$  converts between the world frame and camera frame. A small rotation vector can easily be converted to quaternion, think of a small step  $\delta\alpha \in \mathbb{R}^3$  on a 2D plane mapped to a 3D sphere (Figure 7, Equation 8). This can be altered for optimisation purposes by approximating  $\delta\alpha$  as  $\Delta\alpha$  (Equation 9).

$$\delta \mathbf{q} = \exp \left( \begin{bmatrix} \frac{1}{2} \delta \alpha \\ 0 \end{bmatrix} \right) = \begin{bmatrix} \sin \|\frac{\delta \alpha}{2}\| \frac{\delta \alpha}{2} \\ \cos \|\frac{\delta \alpha}{2}\| \end{bmatrix}^{\oplus} \quad (8)$$

$$\bar{q}^{k+1} = \delta \mathbf{q}(\Delta \alpha) \otimes \bar{q}^k \quad (9)$$

### 3.3 Bicycle States

At each taken image ( $k$ ) the bike can be assumed to have a set state  $\mathcal{X}_R^k$ . Contained within  $\mathcal{X}_R$  are a set of pose states,  $\mathcal{X}_T$ , and speed/bias states  $\mathcal{X}_{sb}$ , Equation 10. Landmarks are contained within the set  $\mathcal{X}_L$ .

$$\begin{aligned} \mathcal{X}_T &:= [{}_{W}r_S^T, q_{WS}^T]^T \\ \mathcal{X}_{sb} &:= [{}_s v^T, b_g^T, b_a^T]^T \\ \mathcal{X}_R &:= [{}_{W}r_S^T, q_{WS}^T, {}_s v^T, b_g^T, b_a^T]^T \end{aligned} \quad (10)$$

The elements  ${}_{W}r_S^T$ ,  $q_{WS}^T$  are the position and the quaternion body orientation,  ${}_s v^T$  the velocity,  $\mathbf{b}_g$  &  $\mathbf{b}_a$  the accelerometer and gyroscope biases. Since  $\delta \mathbf{q}$  describes the axis-angle perturbation about the estimate  $\bar{q}_{WS}$  it follows that  $\mathbf{q}_{WS} = \delta \mathbf{q} \otimes \bar{q}_{WS}$ . Hence the error states combining with (Equation 8) are:

$$\begin{aligned} \delta \mathcal{X}_T &:= [\delta r^T, \delta \alpha^T]^T \\ \delta \mathcal{X}_{sb} &:= [\delta v^T, \delta b_g^T, \delta b_a^T]^T \\ \delta \mathcal{X}_R &:= [\delta r^T, \delta \alpha^T, \delta v^T, b_g^T, b_a^T]^T \end{aligned} \quad (11)$$

### 3.4 The SLAM Problem

If measurements were exact, mapping of an environment would be easy. This, however is not the case, there is often noise associated with each taken measurement and movement. For example distance measurements and IMU readings are subject both to a noise and a bias (calibration error) which can both be assumed to be Gaussian distributed.

$$\tilde{z} = \mathbf{b}_c + s \mathbf{M}_z + \mathbf{b} + \mathbf{n} + \mathbf{o} \quad (12)$$

where  $\mathbf{z}$  is the correct measurement,  $\mathbf{b}_C$  is a long-term constant bias,  $s$  is scaling,  $\mathbf{M}$ - misalignment,  $\mathbf{b}$ - time varying bias,  $n$  - noise, and  $o$  is other un-modelled influences. This means that no measurement can be trusted exclusively for state-space information. When moving therefore, it was first assumed that errors have a compounding effect. The position of a bike, would become increasingly unknown as the environment is explored and the bike is displaced from the origin. This, however, is not the case as proved by Durrant-Whyte [25] since there is a high degree of correlation between estimates of different landmarks and locations in a map, these correlations also increase with successive observations (Figure 10). As an example of this (Figure 10 : Right) displays error in measurements of landmark observation, the mean error is common between all landmarks and so with successive landmark observations the errors in location estimates become highly correlated. This means that although  $\mathbf{m}_i$ 's location may be quite uncertain but the relative location between two landmarks  $\mathbf{m}_i - \mathbf{m}_j$  is almost certain.

The SLAM problem is therefore formulated using both re-projection error (from the camera measurements)  $\mathbf{e}_r$  and an inertial error term from the IMU  $\mathbf{e}_s$ .

$$J(x) := \sum_{i=1}^I \sum_{k=1}^K \sum_{j \in \mathcal{J}(i,k)} \mathbf{e}_r^{i,j,k \ T} \mathbf{W}_r^{i,j,k} \mathbf{e}_r^{i,j,k} + \sum_{k=1}^{K-1} \mathbf{e}_s^k \ T \mathbf{W}_s^k \mathbf{e}_s^k \quad (13)$$

where  $i$  is the camera index,  $k$  the camera frame index,  $j$  the landmark index. Indices of landmarks in the  $k$ th frame and  $i$ th camera are in the set  $\mathcal{J}(i,k)$ . In the case of this project the error is only

estimated with one camera, and the IMU. Any additional sensors used would also need to be included in this error state.

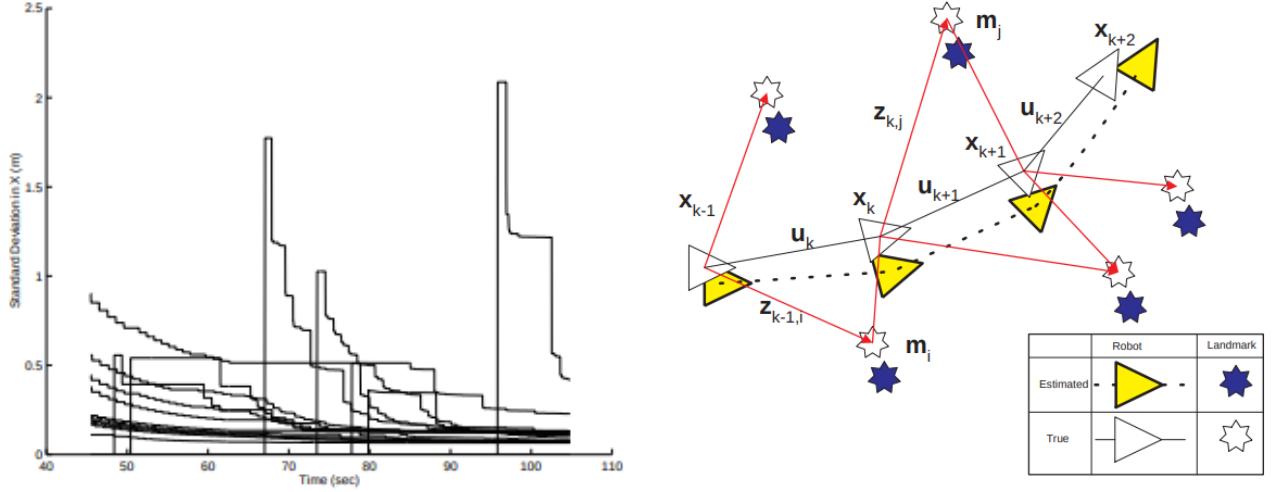


Figure 10: Left: Convergence in landmark uncertainty. Over time, standard deviations reduce monotonically to a lower bound [26]. Right: Error between estimated and true landmarks is common.

### 3.4.1 Camera Re-projection Error

The camera uses a focal-lens to take a photograph. This, especially on cheaper cameras, can distort the image. The distortion is intrinsic to the camera and it needs to be calibrated to determine a distortion model  $\mathbf{h}(\cdot)$ . This is then used to determine the re-projection error [27].

$$\mathbf{e}_r^{i,j,k} = \mathbf{z}^{i,j,k} - \mathbf{h}_i(\mathbf{T}_{CiS}^k \mathbf{T}_{SW}^k w \mathbf{l}^j) \quad (14)$$

where  $\mathbf{z}^{i,j,k}$  denotes the measurement image coordinates.

### 3.4.2 IMU Kinematics

Therefore the kinematics of the IMU can be modelled as:

$$\begin{aligned} {}_W\dot{\mathbf{r}}_S &= \mathbf{C}_{WS}{}_s\mathbf{v}, \\ \dot{\mathbf{q}}_{WS} &= \frac{1}{2}\Omega({}_S\tilde{\omega})\mathbf{q}_{WS}, \\ {}_S\dot{\mathbf{v}} &= ({}_S\tilde{\mathbf{a}} + \mathbf{w}_a - \mathbf{b}_a) + C_{SW}{}_{WG}\mathbf{g} - ({}_S\omega) \times {}_S\mathbf{v}, \\ \dot{\mathbf{b}}_g &= \mathbf{w}_{bg}, \\ \dot{\mathbf{b}}_a &= -\frac{1}{\tau}\mathbf{b}_a + \mathbf{w}_{ba}, \end{aligned} \quad (15)$$

where:

$$\begin{aligned} \Omega({}_S\tilde{\omega}) &= \begin{bmatrix} -S\omega \\ 0 \end{bmatrix}^\oplus \\ {}_S\omega &= {}_S\tilde{\omega} + \mathbf{w}_g - \mathbf{b}_g \end{aligned} \quad (16)$$

${}_W\dot{\mathbf{r}}_S$  represents the position derivative likewise  ${}_S\dot{\mathbf{v}}$  for velocity,  ${}_S\tilde{\mathbf{a}}$  accelerometer measurements,  ${}_WG$  earth's gravitational acceleration vector, and finally  $w := [w_g^T, w_a^T, w_{bg}^T, w_{ba}^T]$  are independent Gaussian white noise processes with zero mean.

### 3.4.3 IMU Kinematics Linearised

In order to optimise the cost function, it is important to linearise the IMU kinematics. The error dynamics take the form:

$$\delta \mathcal{X} \approx \mathbf{F}_c(\bar{x}_R) \delta \mathcal{X}_R + \mathbf{G}(\bar{x}_R) \mathbf{w} \quad (17)$$

$$\begin{bmatrix} \delta \dot{\mathbf{r}} \\ \delta \dot{\mathbf{a}} \\ \delta \dot{\mathbf{v}} \\ \delta \dot{\mathbf{b}}_g \\ \delta \dot{\mathbf{b}}_a \end{bmatrix} = \begin{bmatrix} 0_{3 \times 3} & [\bar{C}_{WS}\bar{v}]^\times & \bar{C}_{WS} & 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & -\bar{C}_{WS} & 0_{3 \times 3} \\ 0_{3 \times 3} & -\bar{C}_{WS}[W\mathbf{g}]^\times & -[S\bar{\omega}]^\times & -[S\bar{v}]^\times & -I_3 \\ 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & -\frac{1}{\tau} \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \delta \mathbf{r} \\ \delta \mathbf{a} \\ \delta \mathbf{v} \\ \delta \mathbf{b}_g \\ \delta \mathbf{b}_a \end{bmatrix} + \begin{bmatrix} 0_{3 \times 3} \\ \bar{C}_{WS} \mathbf{W}_g \\ \bar{C}_{WS} \mathbf{W}_a \\ \mathbf{W}_{bg} \\ \mathbf{W}_{ba} \end{bmatrix} \quad (18)$$

This formulation of the error dynamics is then used to derive the IMU error term (Equation 19), for further details see [24].

$$\mathbf{e}_s^k(\mathbf{x}_R^k, \mathbf{x}_R^{k+1}, \mathbf{z}_s^k) = \begin{bmatrix} {}^W\hat{\mathbf{r}}_S^{k+1} - {}^W\mathbf{r}_S^{k+1} \\ 2[\hat{\mathbf{q}}_{WS}^{k+1} \otimes \mathbf{q}_{WS}^{k+1-1}] \\ \hat{\mathbf{x}}_{sb}^{k+1} - \mathbf{x}_{sb}^{k+1} \end{bmatrix} \quad (19)$$

### 3.4.4 Keypoint Detection

The Okvis [24] paper uses a corner detection method to determine keypoints and then matches these keypoints using a binary descriptor (BRISK [28]). Binary descriptors use a sampling pattern to create pairs of points (lines). The algorithm then compares the two points ( $p_1, p_2$ ) and if  $p_1 > p_2$  a 1 is recorded and otherwise 0. This creates a binary string, that along with orientation representation describe a keypoint. When matching keypoints between images a simple, and fast  $\sum XOR(img1, img2)$  reveals keypoint similarity from which a threshold can be used to determine matches.

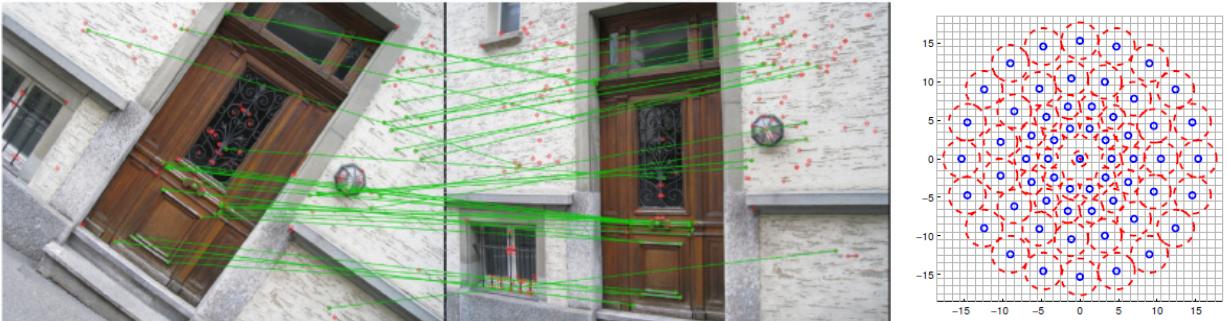


Figure 11: Left: Brisk keypoint matching example, Right: Brisk Sampling Pattern [28]

The BRISK sampling pattern (Figure 11: Right) uses a standard Gaussian smoothing at each sample point. The standard deviation of the sample is represented by the red circles in the figure. This ensures the sample is less prone to noise than other traditional binary detectors e.g. BRIEF.

## 4 Background, Object Tracking

Once an object is found (Section 2) and the 3d world coordinates of the object in question (bike, car, pedestrian etc.) are determined (Section 3) the path of the object needs to be calculated. This requires the ability to track an object from frame-to-frame and transfer that knowledge to 3D space. Mask-RCNN as an instance segmentation solution (Section 2) and runs slower than real-time unless on a powerful GPU system, which cannot be feasibly carried on a bike. So once an object is detected it is preferable to learn simpler features and track it with more basic but faster techniques. An object

from frame-to-frame can be easily distinguished if the features of that object are not occluded and colour balance (for example sunny to shaded regions) is minimal. This, however, is not the case in real-world environments and so the tracking task is less simple than it appears.

Section 3 used the IMU measurements and camera measurements to determine the location of the bicycle and objects in the world frame. This problem is similar but involves the tracking of an object. In [29] there are two parts at play, the camera measurements, and a predictive Kalman filter using the object’s motion information. The Kalman filter creates a bounding box of the expected object location, the object is then correctly located in that region, and the Kalman filter is then updated.

## 4.1 CamShift

Camshift is a modification of the Meanshift algorithm that locates the maximum of a density function. In this case the probability of a region containing the colour and hue values of a previously discovered object in a new frame. It is an iterative algorithm that starts with an initial estimate  $\mathbf{x}$  and a Kernel function  $\mathbf{K}(\mathbf{x}_i - \mathbf{x})$  which determines the weighting of nearby points for mean re-estimation.

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x)x_i}{\sum_{x_i \in N(x)} K(x_i - x)} \quad (20)$$

Where  $N(x)$  is where  $K(x_i \neq 0)$ . This, however does not account for objects moving in a frame and therefore a dynamic scaling approach is needed where the kernel size is variable. Hence CamShift [30] solves these issues by updating the window size every time mean shift converges until a required threshold is reached.

The Meanshift and Camshift algorithms require a probability density to describe objects in each image. Since, as mentioned earlier, colour values in the RGB space can easily alter due to occlusion and other light-variation effects it is suggested in [29] to convert the image to an HSV space before creation of a 48-bin histogram. HSV uses hue, saturation and brightness and the transformation from RGB is as follows:

$$H = \begin{cases} 60 \times \left[ \frac{g-b}{\max(r,g,b)-\min(r,g,b)} \right], & \text{if } r = \max(r,g,b) \\ 60 \times \left[ \frac{b-r}{\max(r,g,b)-\min(r,g,b)} + 2 \right], & \text{if } g = \max(r,g,b) \\ 60 \times \left[ \frac{r-g}{\max(r,g,b)-\min(r,g,b)} + 4 \right], & \text{if } b = \max(r,g,b) \end{cases} \quad (21)$$

$$S = \frac{\max(r,g,b) - \min(r,g,b)}{\max(r,g,b)}$$

$$V = \max(r,g,b)$$

where H, S and V are the hue, saturation, and brightness value of each pixel and r,g,b are red, green and blue values.

## 4.2 Object Tracking Summary

The method discussed in this section uses image segmentation, a predictive Kalman filter with movement information, and the Camshift algorithm. The update step of the Kalman filter requires a measurement, this is obtained from the Camshift algorithm that iteratively and accurately locates the mid-point of a object in the image. This then updates the Kalman filter which predicts where the object will be in the next image and hence dictates where Camshift starts from at step  $k + 1$ . This means that the expensive Mask-RCNN needs to be called far less frequently and hence makes the entire system less computationally expensive. The architecture used in [29] is displayed below (but uses a different image segmentation method of SVMs and Adaboost) in Figure 12.

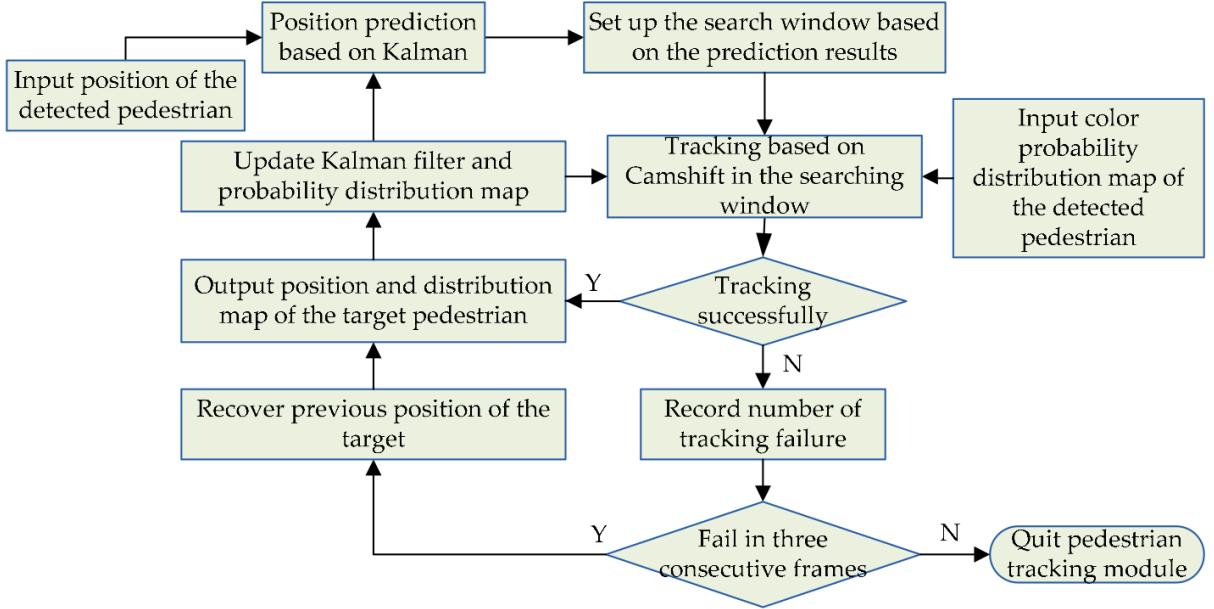


Figure 12: Pedestrian Tracking Module [29]

## 5 ROS

The program is split into respective parts, camera driver, okvis, Mask RCNN and a tracking component. These are all required to communicate for a comprehensive system, hence, since the Mask RCNN implementation used is written in python and okvis plus the driver in c++ ROS is an easy method to transfer messages between nodes. ROS uses a subscriber and publisher methodology where a node can subscribe or publish to a topic. ROS is the de-facto standard library for robotics and allows cross-language compatibility between python and c++.

The ROS library is optimised for development of robotics applications with useful graphing, message passing and debugging capabilities. In particular ROS supports a rosbag capability which allows the recording and playback of data. This should allow for easy testing of the application in less-than real-time as Mask RCNN operates at 0.2s/frame on a dedicated GPU.

## 6 Frame-to-Frame Tracking

For each frame Mask RCNN outputs a set of bounding boxes, and masks. These are not within any particular order and so a method must be devised to track objects from one frame to the next. This is a hard challenge due to object occlusion, variation in shadow/lighting conditions and rapid in-frame movement. An obvious method is intersection over union (IOU) of the bounding boxes on a frame-by-frame basis. This is good in theory, however, it lacks the ability to compensate for rapid movement (of the object and the camera angle), occlusion and object overlapping. A set of methods have therefore been developed to overcome this problem, which combined provide a robust tracking system.

The method must both track objects in the frame, determine which objects are no-longer in-frame and find any new objects that enter the frame. Information about each object must be stored as to determine the velocity and direction of travel/ if a collision is likely between the cyclist and the object in question. Figure 13 displays an example of a working system.

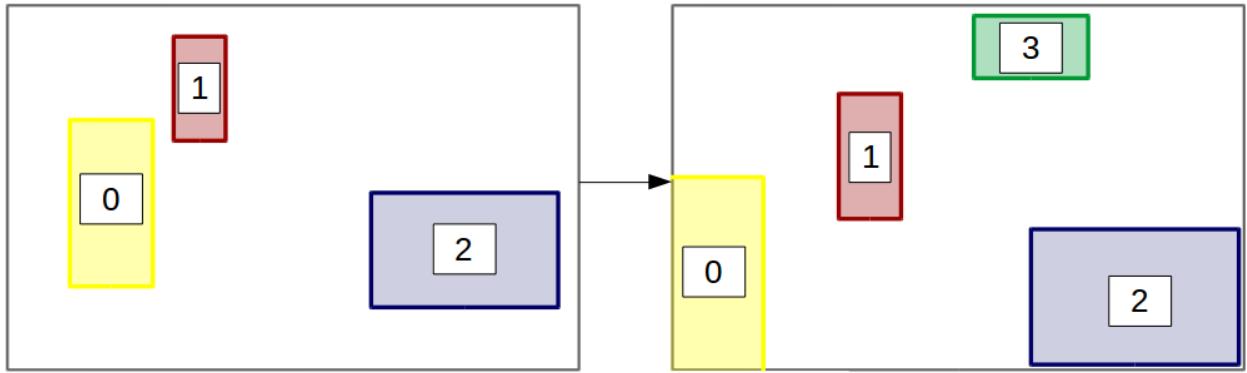


Figure 13: ROI Old to New Frame Example.

### 6.1 Intersection Over Union

Intersection over union (IOU) is a simple method of comparing areas. Consider that there are a number of objects detected in one frame to the next. The majority of these objects can be assumed to have existed in the previous frame, excluding objects that pass out-of-frame and new objects that come into-frame. Hence, assuming a high enough frame rate an IOU score can be computed for each box in the old frame relative to the newer frame boxes. This, with a threshold value can be used to match objects between each frame. The threshold value therefore needs to be established, this will be dependent on frame rate and the relative speed of objects with respect to the camera.

$$\text{IOU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

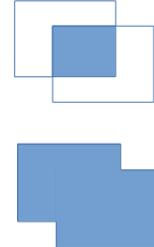


Figure 14: Intersection Over Union Equation.

The worst-case scenario is a person/object moving perpendicular to the camera, the best case directly towards the camera. Sailing has a concept known as constant bearing decreasing range (CBDR) Figure-15 which is often used for collision detection between boats. It is a simple principle that is useful in this case, an object that is a hazard (there is a collision potential) will stay at a roughly constant angle to the bicycle. Hence the only objects we really care about are ones that IOU should be able to track as there is little movement perpendicular to the camera. This being said occlusions must still be dealt with as they may pass behind another object and need to be tracked. It can also not be assumed that the bike will be travelling in a straight line and steering movement will affect the camera angle significantly.

### 6.2 BRISK - Selector

BRISK as described earlier in Section-3.4.4 is a binary descriptor that can match keypoints across two photographs. This can be used also in the tracking section to match keypoints across ROIs, although not thoroughly accurate, it offers a method of matching ROIs regardless of location. This should help re-detect objects after periods of occlusion or overlapping with others that IOU would be unable to do. The process is simple to match the ROIs. First keypoints and descriptors are calculated for each ROI, these are then compared using the fast binary matching process of BRISK, then the Hamiltonian

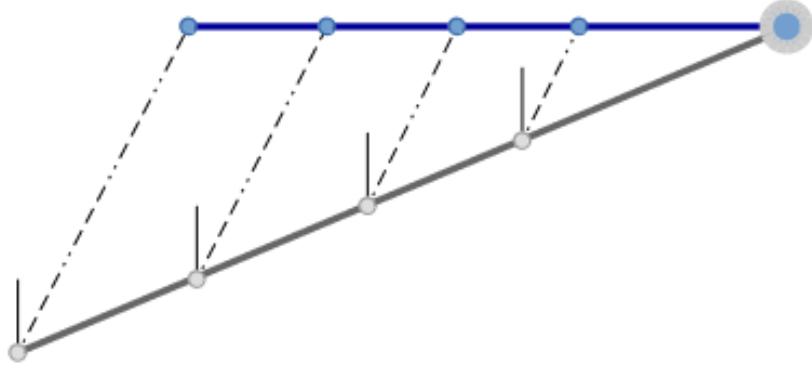


Figure 15: Constant Bearing Reducing Distance.

distance is calculated for each matching keypoint...

### 6.3 Class Matching

An obvious additional requirement is the matching of classes between frames, as a person's ROI should not match to the ROI of a car in the next frame. Hence for each ROI, the class is stored for access by the next frame. This can both reduce expensive computation time of other metrics but provides an easy ROI comparison tool. It can be noted though that Mask RCNN is not perfect and often mis-classifies objects. It is, however, reliable enough to be assumed as correct.

### 6.4 Kalman Filter ROI Matching

The Kalman filter has two main parts, a predictive step and an update. The predictive step tries to find a-priori estimates of the state for the next time-step by projecting forward the current state and error covariance estimates. This estimate is then updated using the measurements (camera) of the object location. The iterative process continues improving the Kalman filter estimates at each stage.

#### 6.4.1 Kalman Predictive Step

Assuming a linear system with Gaussian process, the state can be modelled as:

$$\hat{X}_k = F_k \hat{X}_{k-1} + w_{k-1} \quad (22)$$

where  $k$  is the time step,  $\hat{X}_k \in \mathbb{R}^n$  is the state vector,  $F_k$  is the  $n \times n$  state transition matrix from state  $k$  to  $k+1$  and  $w_k$  is  $\mathcal{N}(0, Q)$ . The state vector is described as follows:

$$\hat{X}_k = [Cx, Cy, w, h, Cx_{vel}, Cy_{vel}, w_{vel}, h_{vel}] \quad (23)$$

where  $(Cx, Cy)$  is the centre point of the bounding box of the object,  $w$  the width and  $h$  the height.  $Cx_{vel}$  describes the centre point velocity in the x direction, and  $Cy_{vel}, w_{vel}, h_{vel}$  the y direction,  $w$  the width and  $h$  the height.  $[Cx, Cy, w, h]$  describe the measurable variables of the system, and there are no controllable parameters in this system. From basic equations of motion, a general measurable parameter  $\gamma$ , and corresponding velocity  $\gamma_{vel}$  follows the update rule:

$$\begin{aligned} \gamma_{k+1} &= \gamma_k + \gamma_{vel} \times \Delta t \\ \gamma_{vel,k+1} &= \gamma_{vel,k} \end{aligned} \quad (24)$$

where  $\Delta t$  is the time step between each frame. Therefore, the update matrix for the filter  $F_k \in R^{8 \times 8}$  is as follows:

$$F_k = \begin{bmatrix} I_4 & \Delta t \times I_4 \\ 0_{4 \times 4} & I_4 \end{bmatrix} \quad (25)$$

The final term  $w_k = \mathcal{N}(0, Q)$  describes the process noise in the system and is normally distributed with error covariance matrix  $Q$ . This is updated by the kalman filter at each step but is initially set as:

$$\begin{aligned} v &= [5, 5, 5, 5, 10, 10, 10, 10] \\ Q_k &= \text{diag}(v) \end{aligned} \quad (26)$$

where  $Q_k \in \mathbb{R}^{8 \times 8}$  and is the diagonalised matrix of  $v$ .

#### 6.4.2 Kalman Update Step

The measurement step can also predict the current state. With  $Z_k \in \mathbb{R}^8$  as the system measurement vector:

$$Z_k = H_k X_k + v_k \quad (27)$$

where  $H_k$  is an  $8 \times 4$  measurement matrix relating  $X_k$  to the measurement  $Z_k$ , and  $v_k$  is  $\mathcal{N}(0, R)$ . Since only the  $Cx, Cy, w, h$  is measurable  $H_k$  takes the form:

$$H_k = [I_4 \quad 0_{4 \times 4}] \quad (28)$$

and the measurement noise  $v_k = \mathcal{N}(0, R_k)$  is set with covariance matrix:

$$\begin{aligned} r &= [5, 5, 5, 5] \\ R_k &= \text{diag}(r) \end{aligned} \quad (29)$$

where  $R_k \in \mathbb{R}^{4 \times 4}$  and is the diagonalised matrix of  $r$ .

#### 6.4.3 Kalman Equations

The prior estimates for the next time step by the Kalman predictions step (no control vector):

$$\begin{cases} \hat{X}_k(-) = F_k \hat{X}_{k-1}(+) \\ P_k(-) = F_k P_{k-1}(+) F_k^T + Q_k \end{cases} \quad (30)$$

where  $\hat{X}_k(-)$  is a priori state estimate and  $P_k(-)$  is the priori estimate error at step  $k$ . It is essentially the previous state  $\times$  the transition matrix to the new state ( $\hat{X}_k(-)$ ). Then a similar approach for the error covariance between the previous and current state. The correction step is then:

$$\begin{cases} \tilde{y}_k = Z_k - H_k \hat{X}_k(-) \\ S_k = R_k + H_k P_k(-) H_k^T \\ K_k = P_k(-) H_k^T S_k^{-1} \\ \hat{X}_k(+) = \hat{X}_k(-) + K_k \tilde{y}_k \\ P_k(+) = (I - K_k H_k) P_k(-) (I - K_k H_k)^T + K_k R_k K_k^T \\ \tilde{y}_{k|k} = Z_k - H_k \hat{X}_k(+) \end{cases} \quad (31)$$

where  $K_k$  is the  $n \times m$  Kalman gain matrix,  $\hat{X}_k(+)$  is the posterior state estimate according to the actual measurement  $Z_k$  and the predicted measurement  $H_k \hat{X}_k(-)$  and  $P_k(+)$  is the posterior state estimate error covariance.

#### 6.4.4 Kalman Matching ROIS

From one frame to another it can be assumed that a number of bounding boxes (ROIS) are consistent in both frames. Hence a metric to match a box from a prior frame to the current frame is required. Using the Kalman filter, a prediction of where the ROI is at time  $k$  is easily obtained. But this does not give a likelihood measure of the boxes in the new frame. However, in the kalman update step

$\tilde{y}_k$ , the innovation residual, and  $S_k$ , the innovation covariance matrix are calculated. These can be used to determine the marginal likelihood of a newer ROI's  $Cx, Cy, w, h$  given the kalman  $P_k(-)$  and  $\hat{X}_k(-)$ . Skipping the formulation, the log-marginal likelihood is:

$$\begin{aligned} l &= \log p(\mathbf{z}) \\ l^{(-1)} &= 0 \\ l^{(k)} &= l^{(k-1)} - \frac{1}{2}(\tilde{y}_k^T S_k^{-1} \tilde{y}_k + \log |S_k| + d_y \log 2\pi) \end{aligned} \quad (32)$$

where  $d_y$  is the dimension of the measurement matrix. The part dependent on the new ROI is  $\tilde{y}_k^T S_k^{-1} \tilde{y}_k$  as  $S_k$ , from Equation 31 does not involve  $\tilde{y}_k$ . Therefore the matching step can be performed by simply matching each  $ROI_{old}$  in the old frame to the  $ROI_{new}$  in the new frame that has the lowest scalar value from:

$$M = \tilde{y}_k^T S_k^{-1} \tilde{y}_k \quad (33)$$

with a threshold value to ensure no incorrect matches. The matching procedure must be the same for all methods and matches the maximum value. Therefore the *Match Metric* is as follows:

$$Match\ Metric = \begin{cases} 0 & if\ K_{threshold} - M \leq 0 \\ K_{threshold} - M & otherwise \end{cases} \quad (34)$$

## 6.5 ROI Lives

Each region of interest may be tracked from one frame to the next. However, if that object is occluded then it may not be spotted in the next frame. This, unless accounted for could lead to information loss regarding that object's state. Therefore a life system is in place whereby an object has a number of lives that are used when the object is not spotted in the next frame. These are decremented until 0 when the object is forgotten and no longer tracked.

## 6.6 Combining Matching Methods

Taking the example provided in Figure 13, there are three objects in the older frame and four in the new frame. BRISK (Section 6.2) is not reliable to match all objects from one frame to the next and sometimes cannot find matching keypoints between the same object in each frame (especially deformable objects such as people). Therefore, a system to combine methods is needed. For each  $ROI_{old}$  and each  $ROI_{new}$  the match metrics for each method (BRISK,Kalman) is calculated. This forms a matrix  $matches \in \mathbb{R}^{new \times old \times methods}$  where  $old, new$  are the number of ROIs in the old and new frames, and  $methods$  is the number of matching methods implemented (in this case two) but is designed to be easily extendible to more. The rules for the matching procedure are as follows:

1. Multiply along *methods* axis to form a  $new \times old$  dimension matrix.
2. Check max element in array is greater than a threshold value.
3. Match new and old ROI of max element.
4. Correct Kalman filter.
5. Set max element value in array to -1.
6. Repeat steps 2-5 until 2 is False.
7. Decrement a life from any old ROIs that are not matched - keep if lives > 0 and predict location using kalman filter.
8. Find  $ID_{max}$  of current objects being tracked.
9. For each new object set ID to  $ID_{max} + 1$  and increment  $ID_{max}$

## 7 Ethics Checklist

	<b>Yes</b>	<b>No</b>
<b>Section 1: HUMAN EMBRYOS/FOETUSES</b>		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
<b>Section 2: HUMANS</b>		
Does your project involve human participants?		✓
<b>Section 3: HUMAN CELLS / TISSUES</b>		
Does your project involve human cells or tissues? (Other than from Human Embryos/Foetuses i.e. Section 1)?		✓
<b>Section 4: PROTECTION OF PERSONAL DATA</b>		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓
<b>Section 5: ANIMALS</b>		
Does your project involve animals?		✓
<b>Section 6: DEVELOPING COUNTRIES</b>		
Does your project involve developing countries?		✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓
<b>Section 7: ENVIRONMENTAL PROTECTION AND SAFETY</b>		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓
<b>Section 8: DUAL USE</b>		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?	✓	

Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓
<b>Section 9: MISUSE</b>		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?	✓	
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		✓
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
<b>Section 10: LEGAL ISSUES</b>		
Will your project use or produce software for which there are copyright licensing implications?	✓	✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
<b>Section 11: OTHER ETHICS ISSUES</b>		
Are there any other ethics issues that should be taken into consideration?		✓

## References

- [1] D. F. Transport, “Table ras30001: Reported road casualties by road user type and severity, great britain, 2016,” report, Dft, 2016.
- [2] D. F. Transport, “Ras50001: Contributory factors in reported accidents by severity, great britain, 2016,” report, Dft, 2016.
- [3] D. F. Transport, “Focus on cycling in reported road casualties great britain 2013,” report, Dft, 2013.
- [4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” vol. 25, 01 2012.
- [7] A. Karpathy, “Networks,” 2017.
- [8] A. Karpathy, “Networks,” 2017.
- [9] A. Karpathy, “Networks,” 2017.
- [10] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” *CoRR*, vol. abs/1710.09829, 2017.
- [11] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *CoRR*, vol. abs/1311.2524, 2013.
- [12] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *CoRR*, vol. abs/1506.02640, 2015.
- [13] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, “Selective search for object recognition,” *International Journal of Computer Vision*, vol. 104, pp. 154–171, Sep 2013.
- [14] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *International Journal of Computer Vision*, vol. 59, pp. 167–181, Sep 2004.
- [15] R. B. Girshick, “Fast R-CNN,” *CoRR*, vol. abs/1504.08083, 2015.
- [16] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *CoRR*, vol. abs/1406.4729, 2014.
- [17] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” *CoRR*, vol. abs/1506.01497, 2015.
- [18] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *CoRR*, vol. abs/1411.4038, 2014.
- [19] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” *CoRR*, vol. abs/1604.01685, 2016.

- [20] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, “Regularization of neural networks using dropconnect,” in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML’13, pp. III–1058–III–1066, JMLR.org, 2013.
- [21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.
- [22] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.
- [23] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask R-CNN,” *CoRR*, vol. abs/1703.06870, 2017.
- [24] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visualinertial odometry using nonlinear optimization,” *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.
- [25] H. F. Durrant-Whyte, “Uncertain geometry in robotics,” vol. 4, pp. 23 – 31, 03 1988.
- [26] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (slam): Part i,” 10 2006.
- [27] P. Furgale, “Extensions to the visual odometry pipeline for the exploration of planetary surfaces,” 01 2011.
- [28] S. Leutenegger, M. Chli, and R. Y. Siegwart, “Brisk: Binary robust invariant scalable keypoints,” in *2011 International Conference on Computer Vision*, pp. 2548–2555, Nov 2011.
- [29] L. Guo, L. Li, Y. Zhao, and Z. Zhao, “Pedestrian tracking based on camshift with kalman prediction for autonomous vehicles,” *International Journal of Advanced Robotic Systems*, vol. 13, no. 3, p. 120, 2016.
- [30] G. R. Bradski, “Computer vision face tracking for use in a perceptual user interface,” 1998.