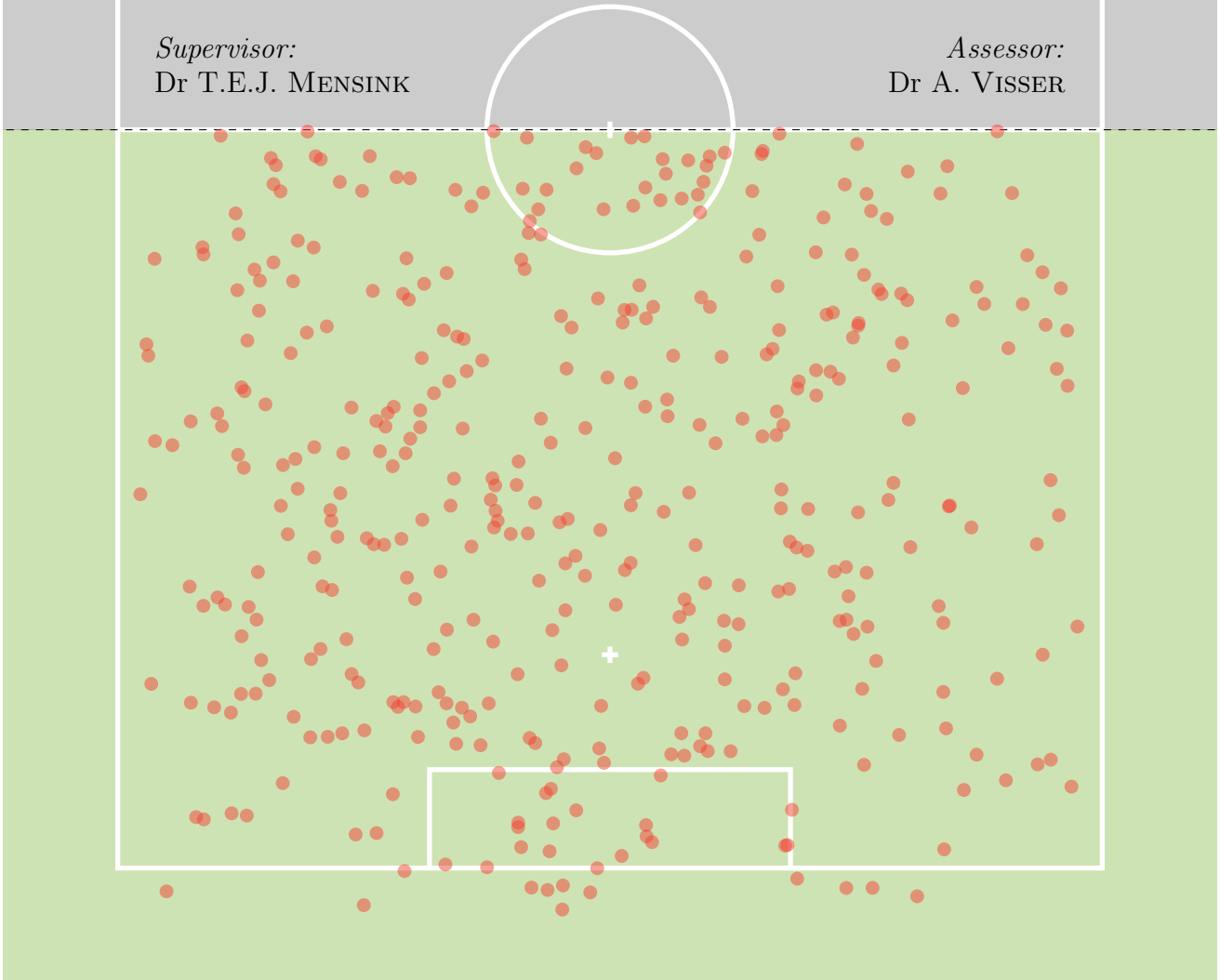
MSc Artificial Intelligence
Master Thesis

Exploiting Symmetries to Relocalise in RoboCup Soccer

Sébastien Negrijn
December 19, 2017

Supervisor:
Dr T.E.J. Mensink

Assessor:
Dr A. Visser



UNIVERSITY OF AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Exploiting Symmetries to Relocalise in RoboCup Soccer

by

Sébastien Negrijn

10340912

December 19, 2017

36 E.C.
January 2017 - December 2017

Supervisor:
Dr T.E.J. Mensink

Assessor:
Dr A. Visser

Graduate School of Informatics

Abstract

In this thesis we are interested in exploiting symmetries in order to more accurately perform relocalisation while playing robotic soccer. For many autonomous robots, localisation is an essential task as without it the next action can not reliably be determined. For example, a robot can not run clear if it has no understanding of its own position in the field, the position of the ball and the position of its teammates and opponents. When the robot no longer knows where it is at it needs to redetermine, or relocalise, its location without being able to use odometry data. A soccer field, fortunately, contains both a point symmetry as well as two line symmetries. By exploiting these symmetries and extending the PoseNet convolutional deep neural net architecture, we use a single RGB image to predict continuous pose coordinates. Instead of generating these images with a real robot, a simulator is created to more easily be able to change parameters such as the lighting conditions, field colour and even the surroundings of the field. After showing that using these symmetries in a soccer field increases the accuracy of the location predictions, we also show an increase when compared to the original PoseNet architecture and its Street Scenes data sets.

Contents

Chapter 1

Introduction

Localisation has been a topic of research in robotics since its origin. During many tasks it is crucial to know where you are to be able to succeed. In the case of this thesis, the task at hand is to play autonomous soccer with a humanoid robot. As per the rules of competition no external sensors can be used leaving the top camera of the robot to be the most suitable sensor. The goal is to use this camera in conjunction with a deep neural net to predict poses from a single image.

1.1 RoboCup

The RoboCup initiative strives to facilitate a publicly appealing goal for robotics and artificial intelligence. It does so by providing various challenges such as logistical difficulties in the *Industrial* track, domestic challenges in the *Home* track, search and rescue challenges in the *Rescue* track and fully autonomous *Soccer*. The logistic difficulties include challenges such as gripping odd sized items with a robotic arm, transporting them to a different location while manoeuvring between obstacles and filtering items if they have defects based on visual inspection. One such example of a logistic challenge can be found in the @Work league where a Kuka YouBot[1] fully autonomous picks up bolts from a moving conveyor belt. Using (depth) cameras, the item needs to be identified and the correct grip position determined, along with the right time at which to grab the item. Based on a previously scanned data matrix (or QR-code), the item needs to be transported to a specific location and put in the correct box. One specific league of the Home track uses the Pepper robot[2] to assist people in domestic situations. Recently, the cocktail party challenge was introduced where Pepper was tasked with getting drink orders from groups of people in a noisy environment, ordering those drinks at the bar and returning the correct drink to the right person. In Rescue, custom build robots are designed to locate people in hard to navigate locations created to simulate disaster sites.

Finally, several soccer leagues exist each focusing on a different main topic of research. As soccer is well known to the general audience and includes many different subfields of artificial intelligence, it is the perfect goal that not only benefits scientists. Computer vision, locomotion and inter agent communication are all relevant fields when trying to play autonomous soccer in which different leagues try to find optimal solutions. The ultimate goal is to play and win a match against the human 2050 FIFA World Cup[3] while obliging to the same rules and without any unfair advantage such as the use of wireless internet to communicate. The *Humanoid Leagues* are based on building humanoid robots that can play soccer without any non-human like sensors like range finders. These leagues are further divided into classes based on height with a maximum height of 90 centimetres for the *Kid Size*, 140 for the *Teen Size* and 180 for the *Adult Size*. Their main research topics include: dynamical walking, running and kicking while still being able to keep their own balance. In the *Middle Size League*, an official FIFA soccer ball is used while only putting restrictions on the height and weight of the robots used. Due to these restrictions, no humanoid robots need to be used and instead wheel based robots are build. As these robots do not need to keep their balance,

[1] http://www.youbot-store.com/wiki/index.php/Main_Page
[2] https://www.ald.softbankrobotics.com/en/robots/pepper
[3] http://www.robocup.org/objective

Figure 1.1: An overview of two teams playing a fully autonomous soccer match in the Standard Platform League at the RoboCup European Open 2016.

and can move omnidirectional, more research can be devoted towards multi-agent cooperation and strategies. The *Small Size League* is the only soccer league that is allowed to use an external camera system to determine the position of the soccer players. The puck shaped 180 millimetres robots can also put most of their effort into multi-agent cooperation as positions of other players and the ball is provided by the external system. Different from the other leagues, the *Simulation League* eliminates the use of robotic hardware as everything is done in simulation. Without the need for often expensive hardware, and only requiring computing power for the simulation new teams can more easily join the RoboCup without first having to purchase robots. Finally, in the *Standard Platform League* (SPL) every team has to use the same robot, creating a fair playing field where the only difference can be made in the software.

1.1.1 Standard Platform League

In the *Standard Platform League* (SPL) every team uses the SoftBank (previously Aldebaran) Nao robot[4] which can be seen in Figure 1.1. Several iterations of this robot are available with only a slight difference in the used gears or available CPU, meaning that every team is limited by the same hardware and the only difference between the teams is in the software. This hardware consists of two RGB cameras in the head where one is aimed at its feet and the other at the horizon. Other sensors include: two sonar beacons and receivers in the chest and several touch sensors and buttons in the head, arms and feet.

The soccer software needs to be run on just the CPU of the robot and no offloading to any other computing power is allowed. This results in the development of well optimised algorithms to detect the ball, other players and the location on the field among others. Typically, this information is shared between the players of one team while playing a match to find the optimal action that should be executed by each player. For instance: as one player is closer to the ball than another, it is most likely beneficial to let the closest player approach the ball. This sharing of information is mostly done using a wireless network, although communication using audio has been on the wish list of some teams due to the unreliability of the wireless network at busy events where multiple of such networks and clients cause interference. As there are plenty of ways to detect objects and share information with other teammates, different teams use different methods which result in different results.

In the SPL, a field is used that is similar to a human soccer field, with a centre line and circle and one penalty box on each side. Recently, a chequered black and white ball replaced the previously used orange ball while in the year before the goals changed from yellow to white making

[4] `https://www.ald.softbankrobotics.com/en/robots/nao`

them harder to detect. These changes make it more difficult to detect the two, while making the field and the surroundings more similar to a human soccer field. As the computing power that is available stays the same, more optimised solutions have to be found. When the goals became the same colour, the field became fully symmetrical as previously the sides of the field could be distinguished by the colours of the goals. It is this symmetry that this thesis will try to exploit while determining the location of the robot. As one can imagine, to be able to score a goal, not only the relative position of the ball, other players, and goal need to be known, but also the position of the robot on the field itself. As in some situations, the goal is not always visible in the current image, the robot would first need to turn towards the goal after which he would walk towards or shoot at the goal. The action of the robot could also be different when the ball is currently close to the defending teams goal as then it might be beneficial to get the ball away from the goal as fast as possible, rather then trying to get it towards the opponent teams goal. Determining the position of the robot on the field is also known as localisation.

1.1.2 Localisation

In general, localisation is considered the determination of the position of an agent within an environment. This can be either a discrete location such as *the opponents penalty box* or *the centre circle*, or specific coordinates relative to some static point. In this thesis we will consider the location of robot relative to the centre of the field, and the orientation around the z axis. The location and the orientation together are considered the pose of the robot. In general, popular sensors for localisation can be divided into two groups: external and internal sensors where the first partially depend on active external sources such as GPS, Bluetooth beacons or WiFi routers which use the time it takes from an external source to the robot itself to determine their relative distance. Using this relative distance to multiple different objects triangulation can be used to determine a relative position. Another recent development is the use of external infrared beacons (lighthouse) that project a known pattern, such as used with the HTC Vive[5], the Microsoft Kinect[6] also projects a pattern but instead is projected from the same location as the camera. The second group often includes range finders that can determine the relative distance in either three or two dimensions between the sensor and objects, depth cameras or ordinary RGB cameras attached to the robot itself. As in the SPL no external sensors can be used, only the internal cameras from the Nao are considered to be useful while trying to determine the position of the robot. From the data that is provided by these internal sensors, usually, features are extracted to reduce the complexity of the data and to provide a more high level description of the data. Furthermore, trends in the data can be used to reduce the uncertainty of the predicted location as the prediction of a pose from a single image might result in several possibilities. A series of images on the other hand, can reduce this uncertainty as more features from several images can be used. This is however not always possible as the stream of images can not always be relied upon as explained next. In the SPL there are two specific cases that have to be dealt with: a fallen robot can have a different orientation after getting back up and a referee might change the position of the robot after it has committed a foul. In this thesis we therefore focus on predicting the pose based on a single image instead of considering multiple images over time.

1.1.3 Relocalisation

Due to the rules of the SPL, situations exist where the robot is picked up by a referee and placed somewhere else. In such cases the internal model that determines where the robot currently is and where the robot could possibly move to in a given amount of time is invalidated. After the model has been invalidated, *relocalisation* is required: without any usable knowledge about previous positions, predicting the current position. The displacement of the robot by an external force is also known as *robot kidnapping* [Engelson and McDermott, 1992] and has been a topic of research for a longer period of time. Similar situations exists outside the SPL as well such as a robot driving into an elevator and ending up on a different floor. A different situation in which a relocalisation algorithm might be required is when the uncertainty from the localisation algorithm has become

[5] `https://www.vive.com/us/product/vive-virtual-reality-system/`
[6] `https://developer.microsoft.com/en-us/windows/kinect/develop`

higher than a given threshold. The detection of when to invalidate the model proves to be difficult even outside of the field of robotics. Similar cases can be found in the field of computer vision when tracking an object over time on sequential images [Smeulders et al., 2014] [Tao et al., 2016]. In this work it is suggested that instead of tracking the object, detecting it in every single image without trying to track it might be easier.

Typically, different algorithms are used for the localisation and relocalisation task when it considers robotics as the former can heavily rely on the odometry, or movement commands, send to the robot to give an estimate of the movement relative to the previous position while the latter has no such information. As relocalisation has to determine the position of the robot in a greater environment than the localisation algorithm, it usually takes significantly longer making the relocalisation algorithm less suited to be executed all the time. Either way, to create and test either a localisation or relocalisation algorithm, image data with known poses is required.

1.2 Data for Robot Relocalisation

To validate an algorithm, usually data with known labels are fed into the network after which the outcome of the algorithm is compared to the known labels. The difference between the predictions and the known value, or ground truth, then provides insight into the accuracy of the developed algorithm. Generating such data in the case of the relocalisation task thus entails having images taken from the top camera of the robot and knowing where those images were taken in the field. Doing this with an actual robot on an official SPL field proved to be challenging.

1.2.1 Learning with the Robot

Applying any research topic to any field that requires hardware and especially robotics is considered to be hard as the robot's servos get hot the longer the robot is in use. One of these topics is reinforcement learning, where an algorithm has to learn by itself what action to perform in certain situations based on previously received rewards. The downside is that to do this, the algorithm needs a lot of attempts or trails to evolve to a solution that is considered to be sensible for every situation. When doing this on a robot that needs to move, the previously mentioned problem occurs causing different results or even a complete malfunction of the robot over time. Another field uses evolutionary algorithms to come to a behaviour that is based on some genome, typically encoded by a series of numbers. Different genomes are generated and tested after which combinations of the best performing genomes are made and the process is repeated. Again, the results from these tests are subject to how worn down the robot is, requiring many breaks between the tests resulting in longer than optimal times required to fully test multiple genomes.

1.2.2 Acquisition of Data from the Robot

During the process of trying to record a data set using an actual robot and a tracking system similar issues were experienced. The used tracking system consists of six OptiTrack[7] infrared cameras from which images are captured. Using these images, reflective markers that were placed on the robot are detected. As the positions of the cameras are known, the detected markers from the images from all cameras can be projected to determine their position in three dimensional space. By combining at least three markers, the pose can be determined as well with an average accuracy less than a millimetre. The cameras were taken to the IranOpen 2017 SPL competitions in Teheran to have access to an official sized field and placed at the corners of the field and ends of the centre line at about three metres high. Each camera was aimed on the centre circle to provide as much coverage as possible. On the robot itself, software was installed to capture images at a set interval and to walk around the field in random directions. Meanwhile, other autonomous robots were actually trying to play soccer while referees were around the field as well to assist them. This setup would have resulted in a data set very similar to an actual match with other moving players, referees and a ball. However, the presence of the referees meant that they were sometimes blocking the view from a camera meaning that not enough cameras were covering that portion of

[7] `http://optitrack.com/products/flex-13/`

Figure 1.2: Example images from the *Virtual KITTI* [Gaidon et al., 2016] data set with real world images on the left and images from a virtual world on the right.

the field to project the detected markers and determine their position. Even when no referee was present at all, the cameras were not able to detect all markers attached to the robot to determine its pose. Even when one of the three markers was not detected, the pose of the robot could not be determined. These issues were most likely caused by having a surface area that was too large for the six cameras that were available to us. Moving the cameras to one half of the field to provide more coverage on a single side was not an option as for the specific experiments that are performed in this thesis, a data set with images on a full field is required.

Several different methods of acquiring the poses of the robot could have been attempted. First additional sensors could have been added to the robot such as GPS. Adding GPS to the robot would however have meant that additional physical hardware had to have been mounted to the robot. Depending on the weight and location, this might influence the balance of the robot while trying to walk and without being able to walk, no realistic data set can be recorded. An additional downside to GPS is that it only provides location data and no orientation. Therefore, in order to determine the orientation a different approach has to be taken such as a compass or *inertial measurement unit* (IMU).

Another method to determine the location and orientation of the robot on the field could be to use an external camera system and instead of detecting markers, detecting the robot itself and its pose. By detecting the robot, the location on the field is known and with the pose, the orientation can be determined. OpenPose[8] [Wei et al., 2016] [Cao et al., 2017] for instance, is a convolutional neural net and is able to detect humans and their poses in real time from RGB images. Preliminary results showed that it was not able to detect the pose of a humanoid robot with any of the pretrained models meaning that images of robots with known poses would have to be gathered to train a specific humanoid robot model.

The solution that was finally chosen for, as is typical in the field of robotics, was to eliminate all hardware, and to instead opt to use a simulation to generate data.

1.2.3 Acquisition and Learning of Virtual Data

The use of simulated data overcomes the many issues that arise when working with sensitive hardware. Due to the recent advancements in computer graphics, generating photo realistic images has proved to be a genuine alternative for actual real world data. The generation of images is only

[8] `https://github.com/CMU-Perceptual-Computing-Lab/openpose`

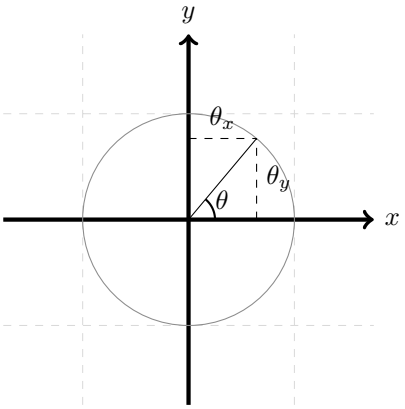


Figure 1.3: Converting an angle (θ) to coordinates (θ_x, θ_y) on the unit circle where $\theta_x = \cos(\theta)$ and $\theta_y = \sin(\theta)$.

one side of the story as being able to generate pixel perfect labels without the need for manual annotation saves both time and resources. Recently, the *Virtual KITTI* data set [Gaidon et al., 2016] was released containing images (Figure 1.2) and videos from outdoor scenes in a virtual world with labels for object detection, optical flow and instance segmentation among others. This research showed that when training a model on the same amount of simulated data instead of real world data, the performance decreases. However, when more data is generated the performance reaches the levels of the model trained on the real world data. This shows that the virtual data contains less information per image but by simply generating more data the same level of accuracy can still be reached when compared to the real world trained model.

In order to generate labelled data for the soccer field the game engine Unreal Engine 4[9] is used. We extend the soccer field simulation[10] of [Fürtig et al., 2017] [T. Hess and Ramesh, 2017] to generate data and labels required to train and test a relocalisation model. Originally, the simulation generated dense (per pixel) labels of which object, such as the ball or a robot, the image pixels belonged to. Each generated sample consists of: an RGB image and a pose. The image consists of 640 by 480 pixels with three colour channels as this is the typical resolution at which the real robot operates. The pose is defined as an x and y coordinate in centimetres relative to the centre of the field and the orientation as the rotation around the vertical z axis. The z coordinate and rotations around the x and y axis are not of our interest as the robots head remains at roughly the same height and the robot remains in a upright position unless it has fallen over. The orientation around the z axis is defined as a coordinate on the unit circle in order to simplify the error function that can be used on the orientation prediction as can be seen in Figure 1.3. If the direct orientation would be used, large errors would have to be corrected at the transition from -180 to 180 degrees whereas no such steps exist in the case of the coordinates.

At each iteration of the generation of a new sample, the robot from which the picture is generated, the ball and the other players are moved to a different random location on the field. As is typical to a real field, the virtual field is illuminated by several lights above the field. As the lights cast shadows from the robots on the field, it might be possible that a trained model could use this information to extract orientation information. As the lighting conditions change per field the simulation also changes the intensity and temperature of the lights for each new generated sample. To further increase the invariance to a specific field, the colour of the field is also changed to various green tones. Initial tests with a data set that had a texture that was based on the real field instead of a simple texture, showed that the model was able to determine which side the robot was on even when no other features were provided. The model was also able to distinguish halves when a surrounding was added to the virtual field such as the Intelligent Robotics Lab . As the goal of our experiments are to exploit the symmetry of the field, the solid colour and a black surrounding are used resulting in the example images which can be seen in Figure 1.4.

[9] `https://www.unrealengine.com/en-US/what-is-unreal-engine-4`
[10] `https://github.com/TimmHess/UERoboCup`

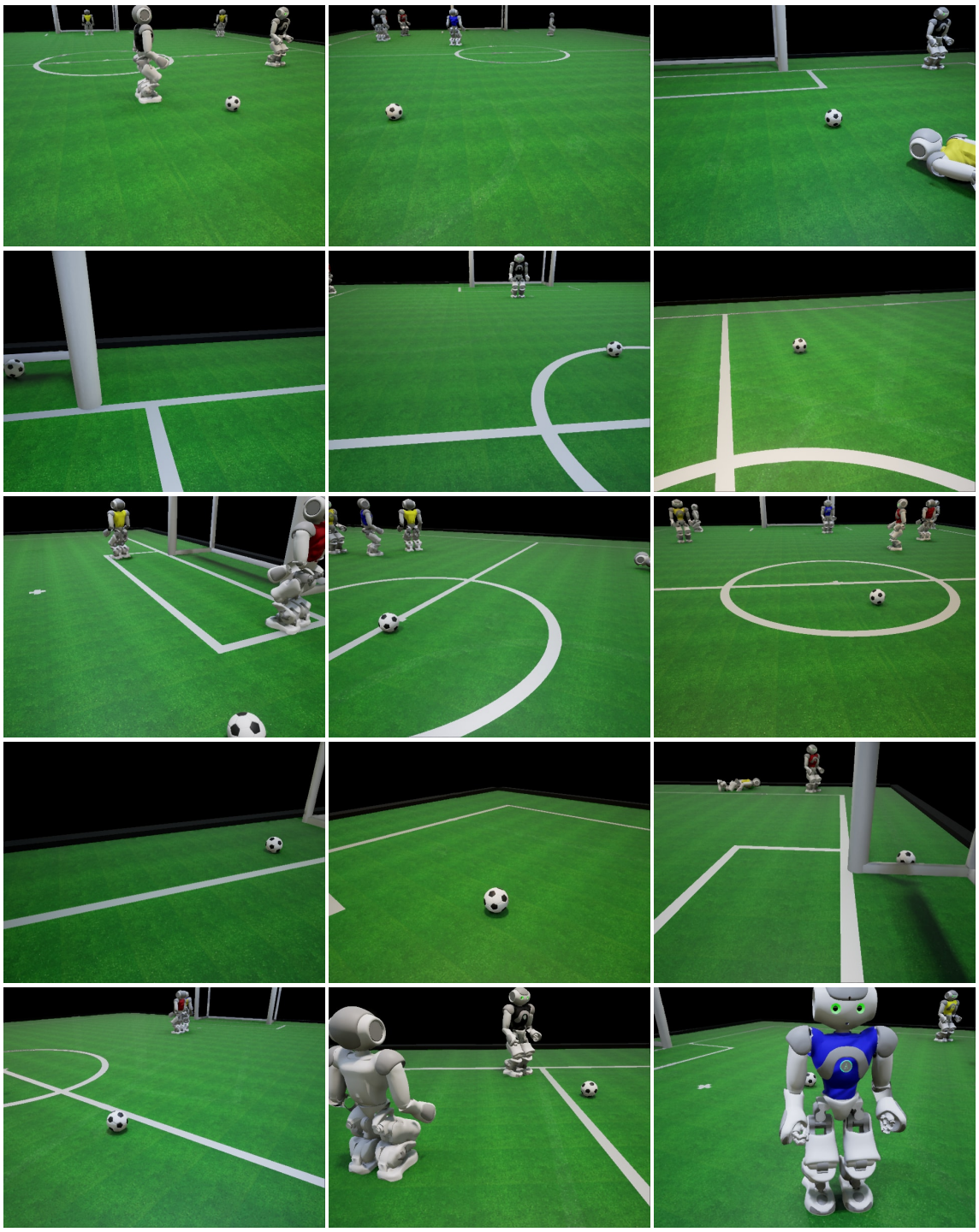


Figure 1.4: Example images from the generated data set in the simulation with different camera view positions, opponent positions, ball positions, lighting intensity and temperature and green field colours.

Chapter 2

(Re)localisation Methods

Various methods exist to determine the pose given a camera image. Two distinct groups can be formed from these methods: based on multiple images or on a single image. The first group uses multiple images to iteratively accumulate the estimated motion between subsequent images. As explained in the previous Chapter, this method is not always suitable due to for instance robot kidnapping. The benefit of this method is that it can be used without any previous knowledge about the environment. This group has great similarity with the second group of localisation methods also known as *relocalisation*. This group is known by this name as they are able to determine the pose of the camera even after a great displacement as they only require a single image. To determine the pose based on a single image, knowledge about the environment is needed to be able to determine the pose. A hybrid between these two methods exists as well known as *Simultaneous Localisation and Mapping* (SLAM) where using the first method, a map of the environment is build at run time and used to localise even after the pose has changed in a unexpected way. That last method has the benefit that no previous knowledge about the environment is required, yet when the robot no longer knows where it is at, it can use the build map to relocalise itself as long as it currently is at a location that it has previously visited.

2.1 Tracking Based

Instead of trying to determine the pose of a robot in an environment from a single image, the movement of the robot can be used to determine its current location based on the previous location. By accumulating the movement of the robot with a given start position, the end position of the robot can be estimated. First the displacement of the pixels in the images must be matched with methods such as optical flow, after which the most likely motion can be estimated.

2.1.1 Optical Flow

One method to determine movement from two subsequent images, is to determine the displacement of the objects, or the optical flow [Horn and Schunck, 1981], in the images. The original paper calculates brightness displacement while assuming that nearby similar valued pixels move smoothly in the same direction, typically features are matched such as Haris corners [Harris and Stephens, 1988] [Nistér et al., 2004] to determine the displacement. More recent work has trained convolutional neural networks named FlowNet [Dosovitskiy et al., 2015], in which different features are first calculated from subsequent images by feeding them into different parts of a siamese network. By feeding them in different feature layers, instead of shared ones, different types of features for the first and second image can be calculated. These features are then fed into a *correlation layer* in which the correlation is calculated between the two feature vectors.

$$c(x_1, x_2) = \sum_{o \in [-k,k] \times [-k,k]} \langle f_1(x_1 + o), f_2(x_2 + o) \rangle \tag{2.1}$$

As simply two features, f_1 and f_2 are compared, no additional training parameters are introduced. The goal of this layer is to provide a measure of how similar the two images from which the features were extracted are as similar features in this case means a similar pose. This layer is then followed by more convolutional and deconvolutional layers to give a dense prediction of the movement. Do note, that the movement of the camera itself has not yet been determined, only the movement of the objects in the image. To determine the movement of the camera, structure from motion can be used.

2.1.2 Structure from Motion

One way to track the position of an agent, such as a robot, is to determine the affine projection of keypoints from an image to a subsequent image over time [Koenderink and Van Doorn, 1991]. In order to do this, first features have to be found in images, typical features include *scale invariant feature transform* (SIFT) [Lowe, 2004] [Wu et al., 2011] [Westoby et al., 2012], *speeded up robust features* (SURF) [Bay et al., 2006], and *Oriented fast and Rotated Brief* (ORB) [Rublee et al., 2011]. These features are used because they are both scale and orientation invariant which is a requirement for features that have to be similar in subsequent images despite the camera of the agent having moved slightly. For image I_t at time point t features F_t are detected. From the image coordinates of the features from F_t three non colinear points are randomly chosen, the first will be the origin of the affine frame and named $\mathcal{O}$ and assigned the coordinates $(0, 0, 0)$, the second $\mathcal{X}$ and $(1, 0, 0)$ and the third $\mathcal{Y}$ $(0, 1, 0)$. $\mathcal{OX}$ and $\mathcal{OY}$ now form the basis vectors of the affine frame. The image coordinates from every other point $\mathcal{P}$ from F_t can be expressed with these two vectors. As the affine transformation happens in a three dimensional space a fourth point $\mathcal{Z}$ has to be chosen as well. This point is assigned the coordinates $(0, 0, 1)$, to find the projection of this point $\mathcal{OZ}$ the line segment, or *trace*, $\mathcal{ZZ}$ where $\boldsymbol{\mathcal{Z}}$ is the same feature from the image I_{t+1} at the next time point, is projected on to the $\mathcal{OXY}$ frame.

The same trick can be applied to any other point $\mathcal{P}$ from F_t where $\boldsymbol{\mathcal{P}}$ is considered the same feature in image I_{t+1}. $\mathcal{P}$ is considered the projection of $\boldsymbol{\mathcal{P}}$ on the $\mathcal{OXY}$ frame and the projection of $\mathcal{P}$ itself. $\mathcal{P}$ is projected on the plane of image I_{t+1} to create a second projected point. The line segment on this plane dived by $\mathcal{ZZ}$ is the third coordinate for the affine projection. This affine projection describes the motion of the camera between images I_t and I_{t+1} and can be repeated for subsequent images.

As only two images are considered, errors from previous affine movements accumulate over time. Furthermore when a robot is kidnapped it is possible that no matches between features from subsequent images are found and that no tracking information can be obtained. In robotic soccer this problem occurs when a player is penalised and manually placed aside the field by a referee. In such a situation relocalisation is required.

2.1.3 Simultaneous Localisation and Mapping (SLAM)

Instead of only keeping track of the movement relative to the previous image, it is also possible to create a three dimensional model of the features that were recorded at each pose and iteratively expand this model with each new image. This method is also known as SLAM [Smith and Cheeseman, 1986] and the subfield that uses ordinary cameras as visual SLAM [Davison et al., 2007]. The advantage of this method in comparison to the ordinary tracking from the previous Section is that error does not necessarily accumulate with each new frame. Several methods have been developed that provide loop closure in the created maps [Newman and Ho, 2005] [Labbe and Michaud, 2014] resulting in high quality maps of the previously unknown environment. As error accumulates while for instance tracking the position of an agent that is driving around a building, after the agent has made a full loop the predicted position will not be exactly the same as the starting position. Fixing this discrepancy is known as loop closure and is done by detecting similar features in the map as the current position after which the constructed map is rectified by a calibration matrix. In the case of this thesis, the field is a given and the creation of a map at run time is not necessary.

2.2 Single Image Based

Instead of tracking the movement over time, the pose can also be determined with just the single current image by comparing it to images with known positions. This can either be done directly, image to image, or by comparing features of those images. Another method extracts a model from the images with the known poses which can then be used to predict the pose for new images.

2.2.1 Parametric Methods

Some methods first build a model with the structure from motion method described above, creating a database of images with known poses also known as *reference images* and *reference poses* [Liang et al., 2013]. Due to the operations of the structure from motion algorithm, it could be that these reference poses are not actual ground truths describing the real world due to the accumulating error. Other ways to obtain pose information include exterior sensors such as GPS and multi camera based tracking systems. For the matching of new images with unknown poses several methods exist to compare them with the reference images such as comparing the fast fourier signatures [Cooley and Tukey, 1965] [Menegatti et al., 2003] [Cassinis et al., 2002], comparing bidirectional texture features [Cula and Dana, 2001] or the already mentioned *scale invariant feature transform* (SIFT) [Lowe, 2004] [Wu et al., 2011] [Westoby et al., 2012], *speeded up robust features* (SURF) [Bay et al., 2006], and *Oriented fast and Rotated Brief* (ORB) [Rublee et al., 2011]. There even exists evidence that small animals use this matching method to localise themselves [Collett et al., 1992].

One inherent problem to these methods is that they assume that there exists a unique one to one mapping of images to different locations. In the real world however, different locations can look very similar such as a seemingly endless corridor with regularly spaced doors. In such cases several reference images with different associated poses would be found as they have a high similarity. The typical way to overcome this problem is to add tracking of the movement of the robot over time. The tracking assumes that the robot can only move a particular distance in a given amount of time, making poses that a far away from the last known position unlikely.

Instead of just assuming a maximum amount of movement, an estimate of the actual movement can sometimes be given from the commands send to the robot also known as the odometry. The predictions from the odometry are then used together with the visual estimations in an extended Kalman Filter [Kalman et al., 1960] or an unscented Kalman Filter [Julier and Uhlmann, 1997] where the extended version does not linearise the modelled mean and covariance.

Another issue is that the accuracy of a system that uses reference images to compare new images with, is that it can only be as good as the number of images in the database. Meaning, that given an image whose pose lies between the poses of reference images, at best, the algorithm would return the pose of the most similar reference image despite the related pose not being the actual pose of the current image. More recent work has adapted to this by first retrieving the most similar images, after which the actual pose of the new image is determined based on the top results [Laskar et al., 2017]. A siamese network is trained to predict the relative camera pose between the two image inputs and as the goal is to capture the geometric properties of such movement, these images can be independent of the scene used at the testing stage. Meaning that images from a different scene can be used at testing time without any retraining. As the trained branches of the network function as feature extractors, the resulting output of a given image can be used to query the database of reference images for most similar images. Then, the new image and the most similar image can be fed into the network to predict their relative camera pose.

2.2.2 Non-Parametric Method

It is however also possible to work without a database and to fully encode the reference images in a convolutional neural network. This network is introduced as PoseNet [Kendall et al., 2015] and predicts full six degree of freedom (DoF) camera poses from single RGB images without the need for any reference images at inference and is the first convolutional neural net to do so to our knowledge. This model uses the GoogLeNet architecture [Szegedy et al., 2015] which consists of nine inception layers as can be seen in Figure 2.1. In turn, one such inception layer consists of four parallel convolutional layers at various sizes to capture different features. The output of these

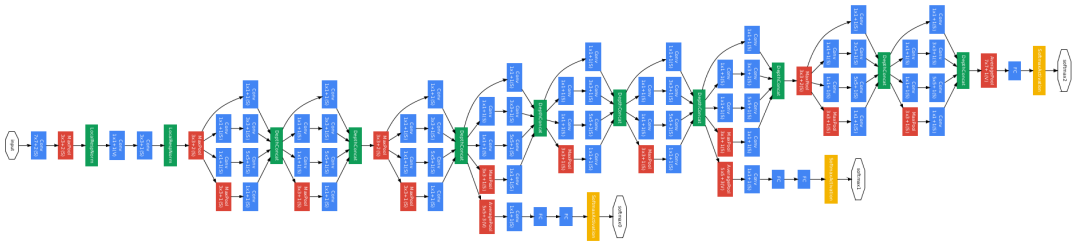


Figure 2.1: An overview of the GoogLeNet architecture used by PoseNet.

layers is then concatenated again after which it is fed into the next inception module. During training of the network, the final regressor layers are repeated at one third and two thirds of the network to partially solve the problem of vanishing gradients which occurs when training a very deep (convolutional) neural network using backpropagation [LeCun et al., 1989]. The network was first trained on the ImageNet data set [Deng et al., 2009] which consists of 3.2 million images from 10184 categories. By first training the network on a large scale data set, the layers generate more general features without overfitting to the training data. The method of first training on a different data set is known as *transfer learning* and typically used when no large data sets of the task at hand are available.

Whereas PoseNet was used to predict the pose of a mobile phone in the city centre of Cambridge, in this thesis the same architecture will be used to predict the pose of a robot on a soccer field.

Chapter 3

Exploiting Symmetries in RoboCup Soccer

The goal of this chapter is to provide an overview of how symmetries in a soccer field can increase the performance of relocalisation. The relocalisation task consists of estimating the pose (x and y coordinates and orientation θ) of the robot on a soccer field. As will be shown, the soccer field is perfectly symmetrical causing uncertainty when prediction the pose as no distinction between the two halves can be made. The model could therefore predict the pose on the other side of the field that, from the point of view from the robot, looks identical to the view from the pose on the correct side.

By explicitly using the symmetry of the soccer field in the encoding of the prediction of the network, we try to isolate this uncertainty in a separate output of the network. Specifically, point and line symmetries are used to create half and quarter field representations. The conversion of the data as well as the training and testing of the various models have all been implemented[1] using the Caffe deep learning framework [Jia et al., 2014].

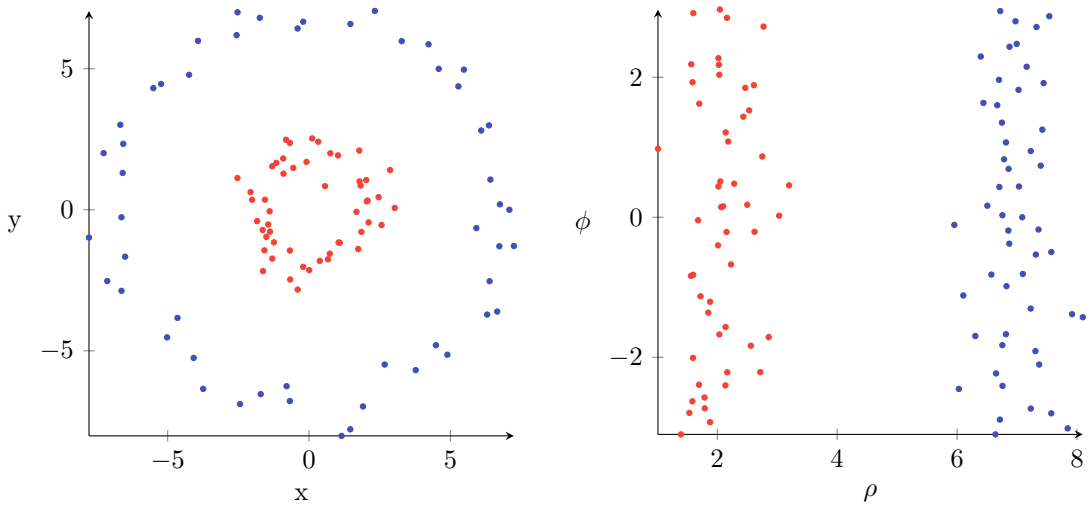
3.1 Detecting Symmetries

Detecting symmetries is something that has been a long topic of research in both the theoretical and the computer science field. Even though the problem is clearly defined due to the definitions of symmetries, it is not a solved problem as is proven by a very recent organised workshop named *Detecting Symmetry in the Wild*[2].

Even though detecting symmetries is not what is done to try to enhance the performance in this relocalisation task, it is a relevant field from which lessons can be learned. For instance from the *RealWorld Images Competition 2013* [Liu et al., 2013], where a submission [Patraucean et al., 2013] uses Hough transform [Duda and Hart, 1972] in addition to *speeded up robust features* (SURF) [Bay et al., 2006] to detect similar features on both sides of a possible symmetry as is a typical method to detect symmetries [Loy and Eklundh, 2006]. And other research [Cicconet et al., 2017] which shows that even better performance can be gained in finding symmetries when using convolutions to create edges, and expecting symmetries to have a mirrored edge across the symmetry. These two submission use two different methods to exploit a symmetry, namely that a similar feature must exist on the other side of the symmetry. They also highlight the, in this case, problem in predicting the correct position of the robot based on a single image. As there are multiple positions that will result in the same image, and these images will result in similar features, finding the correct position based on these features would be impossible.

In the following experiments, the symmetries will be exploited to enhance the performance when trying to relocalise.

[1] The implementation as well as the soccer data set can be found on github: `github.com/S3BASTI3N/robopose`
[2] `https://sites.google.com/view/symcomp17/home`



(a) An example data space plotted as Cartesian coordinates forming two circles.

(b) A feature space plotted after converting the data points to polar coordinates.

Figure 3.2: Transforming from the data space, in this case the Cartesian coordinates, to a feature space, here polar coordinates, can create a linearly separable space from one that was previously not.

3.2 Exploiting Symmetries

Typically in model estimation, data is first transformed into some feature space to ease the learning process. A well known example is the two overlapping circles, as can be seen in Figure 3.2 which can not be linearly separated. After a transforming from Cartesian to Polar coordinates, a simple linear separation is possible.

Instead of transforming the feature space, in this chapter the output space will be transformed. Previous work shows that this can both reduce prediction time and increase performance [Deng et al., 2011]. By instead of creating a classifier that has to predict probabilities for objects in an image to belong to one of 10184 classes in the ImageNet data set [Deng et al., 2009], a tree structure is created with at each node a classifier that determines if the object belongs to the left branch of the current node or the right branch. For example: instead of having a single classifier that has to predict if a depicted animal is a munchkin cat, Siamese cat, German shepherd or a beagle it might be be beneficial to first create a classifier that determines whether the animal is a cat or a dog. Then, the specific species of the cat or dog is determined by two additional classifiers specific to those animals. By determining how many nodes in the tree are added and how many branches those nodes have, a trade off between the accuracy and efficiency of the label tree is made. For the ImageNet data set both an increase of accuracy and efficiency is showed with less training cost when compared to previous created label trees [Torralba et al., 2008] and flat classifiers.

This research shows that by transforming the output space instead of the input space, the performance can be increased as well. Similarly, by posing constrains based on symmetries on the prediction of the location of the robot on a soccer field a similar approach is taken. By separating the prediction of the half the robot is on from the location and orientation of the robot, the cause of wrongly predicted poses should be isolated. In the next Section we will show that a soccer field is symmetric and a several method to encode the symmetry creating two types of half field representations and a quarter field in addition to the full field representations. The encoding also provides the possibility to determine the original pose on the full field by prediction on which half or quarter the robot is on.

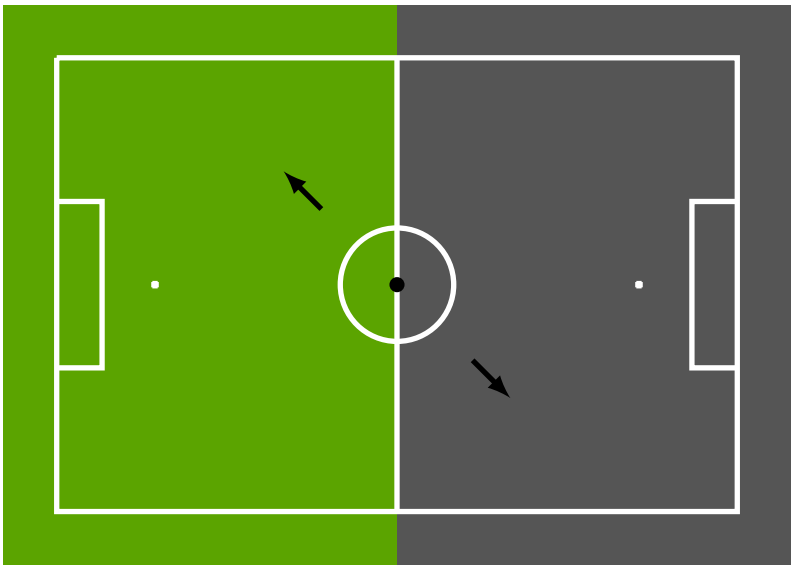


Figure 3.3: Point symmetry where positions on the right side of the field are transformed into positions on the left side by mirroring through the centre circle.

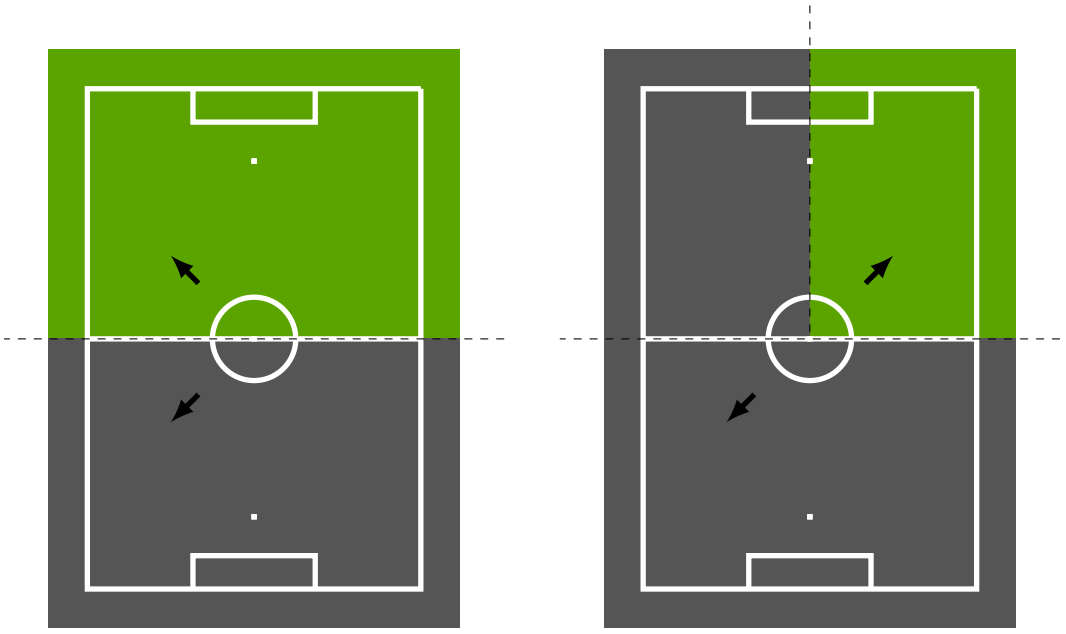
3.3 Symmetries in RoboCup Soccer

An official RoboCup SPL (Section 1.1.1) field consists of a nine by six metres with lines to mark the edges and the middle, as per the rule book of 2017 [RoboCup Technical Committee, 2017]. Each half has a penalty box and a penalty mark and in the centre a circle, similar to an official human soccer field. Typically the field is lined with a five centimetre black border and foam boards with logos of the sponsors or owners of the field. These foam boards break the symmetry but are never the same for any field at any location. Using these boarder features in a deep network would typically mean that the model had to be retrained for every field at every different location resulting in a unpractical situation.

When only considering the field itself however, one can divide the field along the centre mapping the opponents half, onto the home playing teams half as the field itself is perfectly symmetrical. This can be done using *point symmetry*, or *point reflection*, where the centre circle is chosen as the symmetry point. When considering this point, it can be seen in Figure 3.3 that each point on a line segment on the right side of the centre circle has a matching point on the opposite side of the centre circle thus fitting the definition of a point symmetry. When the position of the robot on the field is transformed from the right side to the left side, the resulting produced images from these locations are the same when only considering the field itself. For instance: the penalty box appears to be on the left side of the robot when it is on the right half of field, and this remains the same for the robot on the right side.

Algorithm 1 Converting the data set to the half field representation using the point symmetry.

```
Initialise SymmetryPoses = [...]                    ▷ Initialise array to store converted poses
for all x, y, θ_x, θ_y ∈ Poses do                   ▷ Update each pose
    if y < 0 then                                   ▷ If the pose is on the other half
        h = 1                                       ▷ Other half
        SymmetryPoses ← −x, −y, −θ_x, −θ_y, h       ▷ Append converted Pose
    else
        h = 0                                       ▷ Same half
        SymmetryPoses ← x, y, θ_x, θ_y, h           ▷ Append converted Pose
    end if
end for
```



(a) Half field. **(b)** Quarter Field

Figure 3.5: Using line symmetry to create a half and quarter representation by mirroring along the dashed line or lines.

Another possible symmetry is *line symmetry*, or *reflection symmetry*, as can be seen in Figure 3.4a where positions on the right side of the centre line are transformed into locations on the left side of the line by mirroring along the centre line. Transforming the location of the robot on the field results in a different position on the opposite side when compared the the point symmetry. While only considering the field itself, this would result in a different view to the robot, as the penalty box was previously on the left on the robot, it now appears the be on the right side. This is in contrast to the case with the point symmetry where views stayed the same. Another difference is that in the field, two line symmetries exists while there is only one point symmetry. By applying the two line symmetries the field can be further reduced to a quarter field while this is not the case with the point symmetry. Whether these transformations have influence on the performance of relocalisation is tried in the experiment explained in Section 3.5.

Finally, the field can be further reduced to only a quarter by applying two line symmetries resulting in a field as illustrated in Figure 3.4b. Similar to the single line symmetry, views from the robot flip horizontally but only is some cases. The flip only occurs for positions that do not have to be transformed along the vertical axis. A full performance comparison for the relocalisation task on a full, half and quarter field is given in Section 3.6.

Algorithm 2 Converting the data set to the half field representation using the line symmetry.

```
Initialise SymmetryPoses = [...]                    ▷ Initialise array to store converted poses
for all x, y, θ_x, θ_y ∈ Poses do                   ▷ Update each pose
    if y < 0 then                                   ▷ If the pose is on the other half
        h = 1                                       ▷ Other half
        SymmetryPoses ← x, −y, θ_x, −θ_y, h         ▷ Append converted Pose
    else
        h = 0                                       ▷ Same half
        SymmetryPoses ← x, y, θ_x, θ_y, h           ▷ Append converted Pose
    end if
end for
```

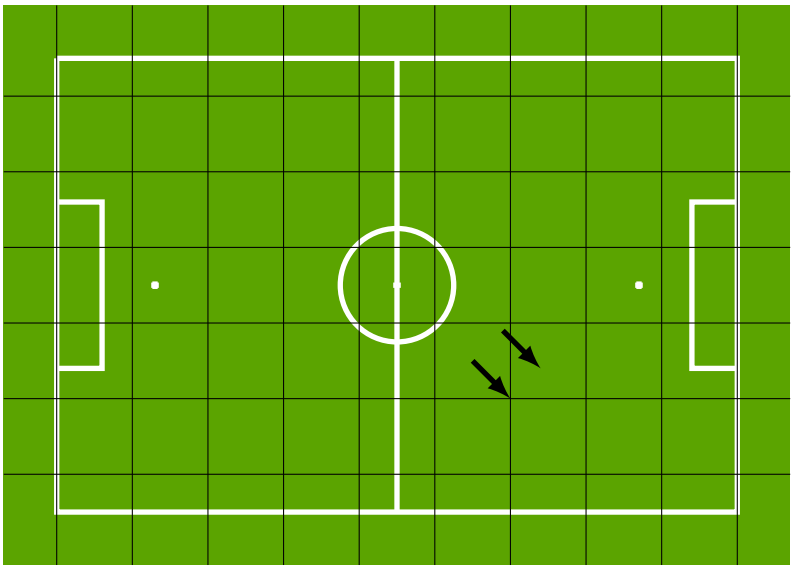


Figure 3.6: Grid wise representation of a robot location where any location of the robot is discretised to the centre of the grid cell it is in.

3.4 Discrete versus Continuous Output Representation

The goal of this thesis is to predict the location and orientation of a robot on a soccer field, to do this in the most efficient way two different location representations are tested: a grid based discrete and a continuous regression representation. The orientation is represented as a coordinate on the unit circle to prevent high error when transitioning from 180 degrees to -180 degrees.

As generally neural networks are considered to be better at discrete predictions than regression [Norouzi et al., 2013], a discrete data set is created from the original data set, by grouping locations into cells of one by one metre. This is done in such as way that the centre circle aligns with the centre of a square as can be seen in Figure 3.6. In order to determine this size we assume that predictions within this one metre box would be considered correct if a binary score for each prediction had to be created. Of course a different size could be chosen depending on the amount of precision needed and the results from a different size might be different from the ones obtained in the experiments done in this thesis. An additional reason for discretising the locations is that for many tasks as a football player, a approximate localisation is sufficient.

Whereas the regression based representation predicts the x and y coordinates of the robot, the grid based method is represented as a one hot vector of grid cells that has a total size of 77 (7×11).

As no longer the exact location of the position of the robot is predicted, the theoretical minimal error increases. To determine this exact number, the positions from the robot are considered to be uniformly distributed within a grid cell. As the error is calculated as the Euclidean distance, the surface of a circle with a radius equal to the average error should be equal to half the size of the grid cell surface. The average error is then calculated as follows:

$$\begin{aligned} \pi r^2 &= \frac{1}{2} w^2 \\ r &= \sqrt{\frac{w^2}{2\pi}} \end{aligned} \tag{3.1}$$

Where r equals the radius of the circle and thus the average error and w the width and height of a grid cell. With a grid size of one by one metres ($w = 1$) it follows that the average error from the grid centre is equal to 0.400. The goal of this experiment is to determine if reducing the problem of finding the location or pose (p) of the robot, which is a x, y coordinate, to a discrete problem, grid cell, will increase the performance even though the resolution at which a correct prediction can be generated will be reduced.

3.4.1 Method

To compare the regression and discrete methods, two models are created using the GoogLeNet architecture, which is described in Section 2.2.2 followed by regression layers. Both models predict the orientation of the robot the same: the feature vector (F) that is produced is multiplied by a single matrix (M_θ) to predict the orientation of the robot.

$$\begin{bmatrix} F_0 & \dots & F_{1023} \end{bmatrix} \times \begin{bmatrix} M_{\theta_{0,0}} & \dots & M_{\theta_{0,1}} \\ \vdots & \ddots & \\ M_{\theta_{1023,0}} & & M_{\theta_{1023,1}} \end{bmatrix} = \begin{bmatrix} \theta_x & \theta_y \end{bmatrix} \tag{3.2}$$

For the regression model, the feature vector is multiplied by a different matrix (M_{xy}) to produce the prediction for the location.

$$\begin{bmatrix} F_0 & \dots & F_{1023} \end{bmatrix} \times \begin{bmatrix} M_{xy_{0,0}} & \dots & M_{xy_{0,1}} \\ \vdots & \ddots & \\ M_{xy_{1023,0}} & & M_{xy_{1023,1}} \end{bmatrix} = p = \begin{bmatrix} x & y \end{bmatrix} \tag{3.3}$$

While the discrete model needs to produce a probability distribution of the probability of the robot being in a certain cell, this is done with a multiplication (M_d) followed by a softmax activation (S).

$$S\left(\begin{bmatrix} F_0 & \dots & F_{1023} \end{bmatrix} \times \begin{bmatrix} M_{d_{0,0}} & \dots & M_{d_{0,1}} \\ \vdots & \ddots & \\ M_{d_{1023,0}} & & M_{d_{1023,1}} \end{bmatrix} \right) = \begin{bmatrix} d_0 & \dots & d_{76} \end{bmatrix} \tag{3.4}$$

$$S(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{3.5}$$

In order to train these networks a loss function has to be defined for the models, which results in the following two equations for the regression model and the grid based models respectively:

$$\mathcal{L}_{reg}(p, \theta) = \alpha \|\hat{p} - x^T\|^2 + \beta \|\hat{\theta} - \theta^T\|^2 \tag{3.6}$$

$$\mathcal{L}_{grid}(g, \theta) = \alpha \mathcal{C}(\hat{g}, g) + \beta \|\hat{\theta} - \theta^T\|^2 \tag{3.7}$$

For the regression model, the euclidean distance is taken between the label, indicated with $\hat{p}$ and $\hat{\theta}$ while for the discrete version the cross entropy $\mathcal{C}$ is calculated for the location instead. To balance the loss between the orientation and location α and β are used. As the bounds for the error for the orientation are lower (0-2) while the location error can increase to the field size (0-1082), the values for α and β are therefore set to 2 and 750 to balance this. A full grid search on various parameter pairs is tested in the next Sections. The network is trained for 12.000 iterations on a train set of 1600 image and label pairs. In order to produce comparable results, for the discrete representation the difference between the location on the field of the grid cell with the higher probability is compared to the label resulting in an error in centimetres. In addition of a direct comparison of the predictions of the models for the half representation, another comparison where the half prediction is used to cast the predicted location and orientation back to the full field representation is given. The first we mark as the oracle models and the second as the modelled ones.

3.4.2 Results

An overview for both the average and median errors, indicated with $\overline{x}$ and $\tilde{x}$ respectively, for the location, orientation and half for the continuous and discrete representations can be found in Table 3.1. A visualisation of the correlation between the error for the location and orientation prediction is given as well in Figure 3.7.

From the results we observe that the location predictions improve for both the discrete and continuous representations for the oracle models after conversion to the half field representation.

Field Type	$\overline{p}$	$\tilde{p}$	$\overline{\theta}$	$\tilde{\theta}$	$\overline{h}$	$\tilde{h}$
Oracle						
Discrete Full	313.85	293.37	4.97	3.46	-	-
Discrete Half	113.29	72.16	20.43	5.91	-	-
Continuous Full	220.83	217.41	45.70	41.95	0.5	0.5
Continuous Half	72.56	44.45	59.75	44.99	0.5	0.5
Modelled						
Discrete Half	291.22	240.12	47.24	20.41	0.5	0.5
Continuous Half	278.49	172.28	91.14	94.00	0.5	0.5

Table 3.1: Results for the regression and discrete representation on full and half fields where p represents the location error (centimetres) and θ the orientation (degrees). In the oracle models the predicted location and orientation are directly compared to the labels where as for the modelled version the half label is used to cast the predictions back to the full field representation.

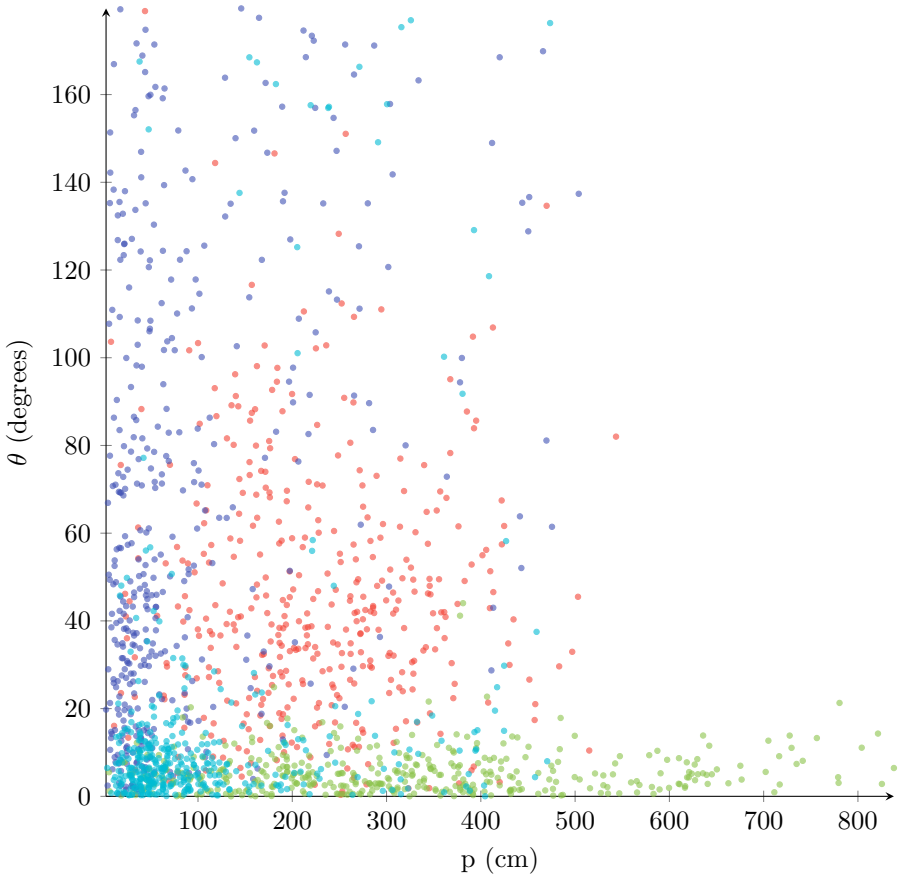


Figure 3.7: Showing the spread and correlations of the error of the full regression (red), half regression (blue), full discrete (green) and half discrete (cyan) representations.

The predictions for the orientation however decrease. When comparing the discrete and continuous half representations the continuous representation performs better on the location prediction while the discrete representation does on the orientation. As the continuous representation provides higher performance on both the half and full field for the location prediction we chose to continue using this representation for the following experiments and to no longer consider the discrete representation. The predictions of the side of the field (h) are almost random, and when used to transform the prediction back to the full field would result in similar performance as the full field representation. The models that use the half prediction to cast back to a full representation perform worse or similar to their full representation versions due to the half prediction being unreliable with a average error of 0.5.

3.5 Point Symmetry on Full and Half Field

In this experiment the point symmetry in the field, as explained in Section 3.3, is utilised and compared to relocalisation on the full field. Because the uncertainty of the half the robot is on is diverted from the location and orientation prediction to the half prediction, a more precise location and orientation prediction is expected.

3.5.1 Method

To predict the orientation of the robot the same architecture is used as described in Equation 3.2, 3.3 and 3.6 for the orientation, location and loss. For the predictions made for the half field however, the side of the field has to be considered as well in the loss function.

$$\mathcal{L}_{reg}(x, \theta, h) = \alpha\|\hat{x} - x^T\|^2 + \beta\|\hat{\theta} - \theta^T\|^2 + \gamma\|\hat{h} - h\|^2 \tag{3.8}$$

The values for α, β and γ are determined by using grid search on a separate test set of the generated data set. To predict which side the robot is on, the feature vector is multiplied by a single matrix.

$$\sigma\left(\begin{bmatrix} F_0 & \dots & F_{1023} \end{bmatrix} \times \begin{bmatrix} M_{h_0} \\ \vdots \\ M_{h_{1023}} \end{bmatrix}\right) = h \tag{3.9}$$

As we want to predict a single probability for the half, the sigmoid function is used to limit the predicted values to values between zero and one.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.10}$$

To ensure that the symmetry property holds true, a generated data set of a soccer field with a black background is used. The final evaluation on the half field will be given in two ways: euclidean distance between the predicted value and the actual value as if there were only one half. The other method will use the half prediction to transform the prediction back to the predicted half and calculate the euclidean distance from this position.

3.5.2 Results

First the values for α, β and γ have to be determined using grid search, of which the average and median errors can be found in Table 3.2. Using these results, the values 1 and 500 were chosen for α and β for the full field and 100, 250, 100 for the half field generating the results on the validation set as can be seen in Table 3.3 and Figure 3.8. The location prediction is greatly improved when the field is reduced to the half field representation as expected. The prediction of the orientation on the other hand, decreases in performance to a level where it is no longer useful as predicting the orientation wrong by an average of 57 degrees results in greatly different result when trying to shoot the ball towards the position where the goal would be based on this orientation.

Field	α	β	γ	$\overline{p}$	$\tilde{p}$	$\overline{\theta}$	$\tilde{\theta}$	$\overline{h}$	$\tilde{h}$
Full	1	500	-	258.56	272.56	8.28	5.33	-	-
	2	750	-	256.99	272.45	9.99	5.62	-	-
	100	250	-	270.28	271.36	11.33	8.81	-	-
	250	100	-	267.14	279.76	15.51	10.37	-	-
Half	100	250	250	88.36	52.96	72.27	56.43	0.5	0.5
	250	100	250	150.36	139.78	96.82	90.77	0.5	0.5
	250	250	100	72.70	63.99	79.91	73.31	0.5	0.5
	100	100	250	77.72	45.64	75.12	68.53	0.5	0.5
	100	250	100	79.90	50.48	59.98	44.77	0.5	0.5
	100	500	100	38.45	84.44	65.75	75.60	0.5	0.5
	100	1000	100	55.93	91.07	52.91	61.26	0.5	0.5
	250	100	100	80.71	52.93	71.66	69.36	0.5	0.5

Table 3.2: Average ($\overline{p}$ and $\overline{\theta}$) and median ($\tilde{p}$ and $\tilde{\theta}$) errors in centimetres for the location and degrees for the orientation, for the point symmetry to create half a field compared to a full field to determine the values of α, β and γ.

Field	$\overline{p}$	$\tilde{p}$	$\overline{\theta}$	$\tilde{\theta}$	$\overline{h}$	$\tilde{h}$
Full	252.18	254.62	4.79	6.99	-	-
Half	82.56	50.28	56.69	42.12	0.5	0.5

Table 3.3: Average ($\overline{p}$ and $\overline{\theta}$) and median ($\tilde{p}$ and $\tilde{\theta}$) errors for the point symmetry to create half a field compared to a full field.

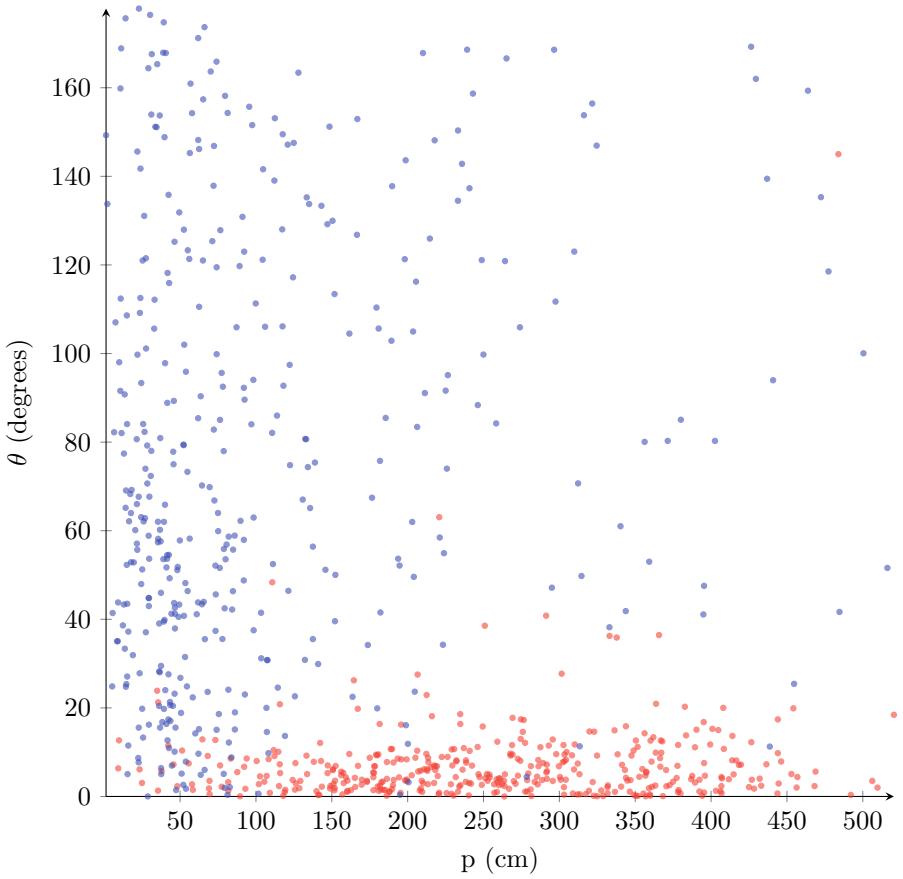


Figure 3.8: Errors for both the location p (centimetres) and orientation θ (degrees) predictions with the full (red) and half (blue) representations created with a point symmetry on the validation set.

3.6 Line Symmetry on Full, Half and Quarter Field

Similar to the previous experiment, but this time with line symmetry, a full, half and also a quarter field are tested. As explained in Section 3.3, transforming the pose of the robot across the line symmetry causes the image the flip horizontally while this is not the case when using a point symmetry. On the quarter field however, the horizontal flip occurs only when the pose that is transformed to the destination quarter could also be reached by a single mirror across the centre line. The expectation is that due to this occasional horizontal flip of the image, the performance on a quarter field will not be better than on the half field.

3.6.1 Method

For the full field Equation 3.6 is used as the loss function, while the half field uses the function as defined in Equation 3.8. For the quarter field another variable (q) is introduced to predict if the robot is on the top or bottom quarter of the field.

$$\mathcal{L}_{reg}(x, \theta, h, q) = \alpha\|\hat{x} - x^T\|^2 + \beta\|\hat{\theta} - \theta^T\|^2 + \gamma\|\hat{h} - h\|^2 + \gamma\|\hat{q} - q\|^2 \tag{3.11}$$

Values for α, β and γ are taken from the previous experiment (100, 250, 100). The quarter prediction q is computed with a same shaped matrix as the half prediction h in conjunction with a sigmoid activation to limit the possible values between zero and one.

$$\sigma\left(\begin{bmatrix} F_0 & \dots & F_{1023} \end{bmatrix} \times \begin{bmatrix} M_{q_0} \\ \vdots \\ M_{q_{1023}} \end{bmatrix}\right) = q \tag{3.12}$$

3.6.2 Results

The average and median error for the line symmetry on a half and quarter field are compared in Table 3.4 while the complete distributions of errors in visualised in Figure 3.9. This visualisation shows that the error for the location prediction decreases for both the half and quarter representations when compared to the full representation. The prediction of the orientation on the other hand still appears to be as scattered across the axis when comparing both the half and quarter representations to the full representation. For all representations there appears to be no correlation between the error for the prediction of the location and the orientation.

3.7 Conclusions

From the results it can be seen that for both the continuous and discrete representation, the half field representation significantly decreases the mean and median error for localisation. When comparing the continuous and discrete representation on both the full and half field, the continuous representation performs better than the discrete one on the localisation task while the opposite is true for the orientation. Additionally, the errors for the half on which the image was taken, appears to be near random. This can be explained by the lack of features in the data set that could indicate on which side the robot is on. As the prediction of the location does improve, isolating the half to a separate prediction is successful, but only when this prediction is not used to transform the pose back to the full field.

For the point symmetry the results show how important it is to perform optimisation on the hyperparameters. Furthermore, using the best performing parameters the half field representation outperforms the full field representation on the localisation task. The orientation however shows an increase in error which is counter intuitive. The expectation was that when half a field was used as a label, the identical view, and therefore the opposite orientation, were no longer considered.

When comparing the results for the average location errors from the line symmetry, reducing the output space to a quarter field further reduces the average error. Despite the error for the orientation to increase for the orientation with the half field representation, it decreases again for the quarter field although still being higher than the full field representation. From this the

Type	$\overline{p}$	$\tilde{p}$	$\overline{\theta}$	$\tilde{\theta}$	$\overline{h}$	$\tilde{h}$	$\overline{q}$	$\tilde{q}$
Full	252.18	254.62	4.79	6.99	-	-	-	-
Half	73.07	55.92	81.42	92.61	0.5	0.5	-	-
Quarter	40.92	26.40	68.93	53.35	0.5	0.5	0.5	0.5

Table 3.4: Average ($\overline{p}$, $\overline{\theta}$, $\overline{h}$ and $\overline{q}$) and median ($\tilde{p}$, $\tilde{\theta}$, $\tilde{h}$ and $\tilde{q}$) errors for the line symmetry after creating a half and quarter field.

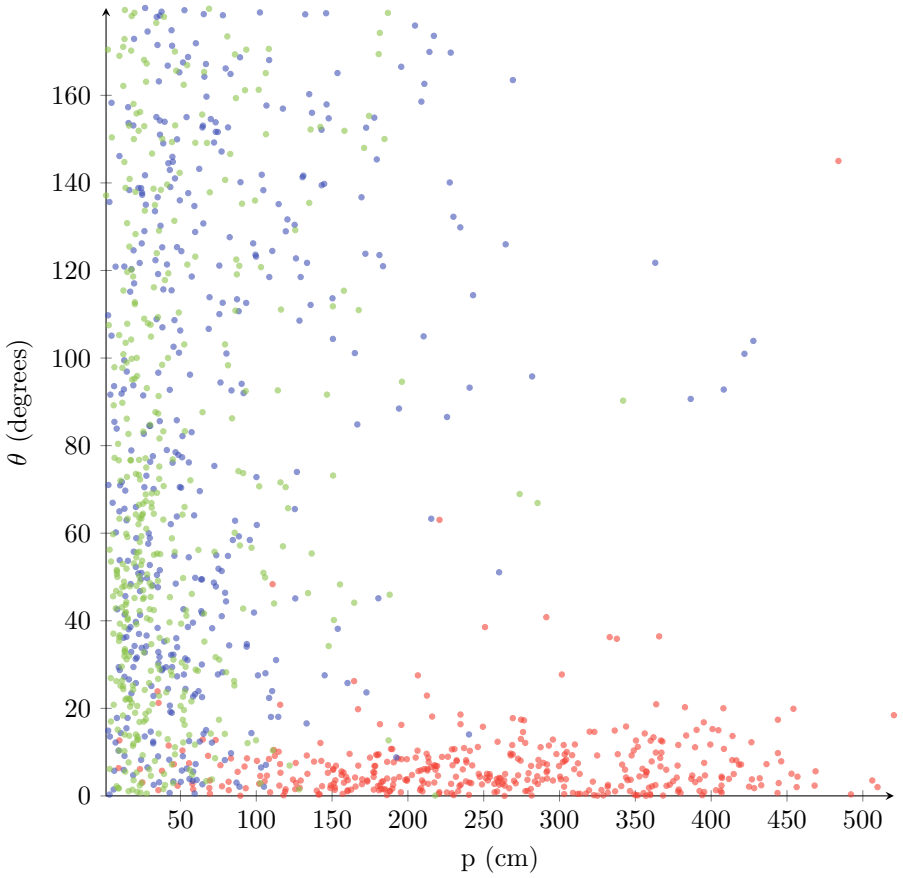


Figure 3.9: Errors for both the location (centimetres) and orientation (degrees) predictions for the full (red), half (blue) and quarter (green) representations created with a line symmetry on the validation set.

conclusion can be drawn that when trying to localise on a soccer field, the best representation to use would be the quarter one but only when another classifier predicts accurately predicts the correct quarter possibly based on features outside of the field.

3.7.1 Line versus Point Symmetry

Finally, the two half field representation that are created with the line and point symmetries are compared from results in Tables 3.4 and 3.3 respectively. The difference in results for the localisation predictions show no significant difference when comparing the point to the line symmetry: 254.62 versus 221.34 on the full field and 50.28 versus 55.92 on the half field. While the difference in orientation does: 6.99 versus 25.40 on the full field and 42.12 versus 92.61 on the half field. This might follow from the horizontal flip in the image that was explained in Section 3.6 even though it seems to not impact the prediction of the location. Therefore, when restricted to a half field representation, using the point symmetry would work best.

Chapter 4

Symmetry in Street Scenes

The experiments in the previous chapter showed that, on an artificial data set with a perfect symmetry, transforming the output space increased the performance. In this chapter, three data sets with ground truth labels of outdoor environments from the original PoseNet model [Kendall et al., 2015] are tested to validate if the proposed approach works in situations where no perfect symmetry is present. These sets are created by walking around the city of Cambridge while recording a video with a mobile phone. Images are taken from this videos and poses are generated afterwards by applying structure from motion, Section 2.1.2, to subsequent images and assumed to be ground truth.

In comparison to the soccer data set, no perfect symmetry is present as there are always additional features that disambiguate the two sides of the symmetry as will be shown in the next Section. Due to these features, a smaller increase in performance is expected when compared to the soccer data set as no such features were present there. First however, the method of applying the symmetry to the three data sets is explained, followed by the results and finally the conclusions that can be made using these results.

4.1 Street Scenes

In comparison to the soccer field, the following data sets do not provide a perfect symmetry, meaning that there is no symmetry point or line across from which the same feature can always be found. They do however contain weak symmetries as will be explained in the following Sections.

4.1.1 Kings College

The first real world data set is composed of images from the King's College entrance. All images depict either the gate itself or the wall attached to it while never showing the other side of the street. As can be seen in Figure 4.1a, the gate appears to contain a symmetry line across the centre, where the left side of the gate is a mirror of the right side (or the other way around).

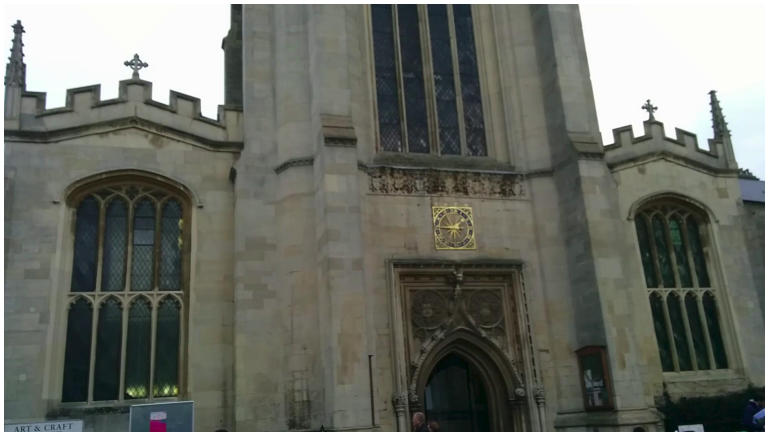
This imperfect symmetry is used to transform the labels of the data set into a data set that has all positions of the camera on one side of the gate, while adding a separate binary label for the side of the camera relative to the gate, similar to the line symmetry experiment on the soccer field explained in Section 3.5. The resulting positions after this transformation are visualised in Figure 4.3a. As the poses in this data set move along a wall, a clear mapping can be made by applying the symmetry resulting in the expectation of an increase in the performance while relocalising.

4.1.2 St Marys Church

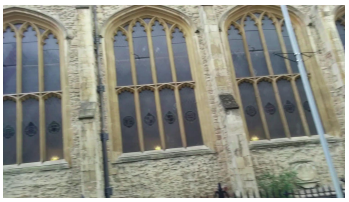
While the previous data set contains images of a relatively flat facade, the next data set is recorded by walking around the Great St Marys Church. From Figure 4.1c, an imperfect symmetry line across the entrance can be seen. Even both sides appear to be similar when comparing Figures 4.1e and 4.1d supporting the symmetry line across the entrance in the front. As in the previous experiment, the symmetry line is used to transform all positions to one side of the church while

(a) View of gate of King's College.

(b) Overview of the King's College walls.



(c) The front view of the St Marys Church.



(d) North (left) view.

(e) South (right) view.

(f) Example images from the Street data set showing that the two sides of the street do not appear the same and no symmetry is present.

Figure 4.2: Example images from the Kings College, St Marys Church and Street data sets.

adding a label that indicates the original side of the church. Even though the poses are not along a straight wall like the previous data set but around an object, a clear mapping can still be made as can be seen in Figure 4.3b. Due to this mapping, an increase in performance while relocalising is expected.

4.1.3 Street

Finally, a data set of a street is used as well. This data is transformed differently from the previous two data sets as no clear symmetries appear in these images as on side of the street does not look the same as the other side, as can be seen in the example image depicted in Figure 4.1f. Instead of prediction the x and y coordinates of the street relative to the starting point of the recording of the data set, the focus is put on how far down the street the image is taken. From Figure 4.3c, it can be seen that the y axis contains relatively small variation when compared to the x axis, we therefore define how far down the street we are as the x coordinate. The y coordinate is still predicted but the positions are normalised by subtracting the mean of the surrounding forty samples. After this, similar to how the line symmetry is enforced, a symmetry across the x axis is assumed, and a label for when a position is on the opposite side of the symmetry is added resulting in the labels as depicted in Figure 4.4. While the output space of this data set is greatly transformed to be able to enforce a symmetry that is not present, no significant improvement is expected. instead, the results from this data set are used to depict the impact of transforming the output space on the performance.

4.2 Method

As was done in the previous chapter, the GoogLeNet architecture is used followed by fully connected layers to predict the orientation represented by a unit circle (Section 1.3) and the location p as xy coordinates (Section 3.4) for the original data sets. The precise architecture for both GooLeNet and the regression layers are explained in detail in Sections 2.2.2 and 3.4. As the trade off between training the location, orientation and half predictions must be found, a test set from the data set is used to determine the hyperparameters α, β and γ. The final performance is then tested on a validation set which has no overlap with either the training nor the test set.

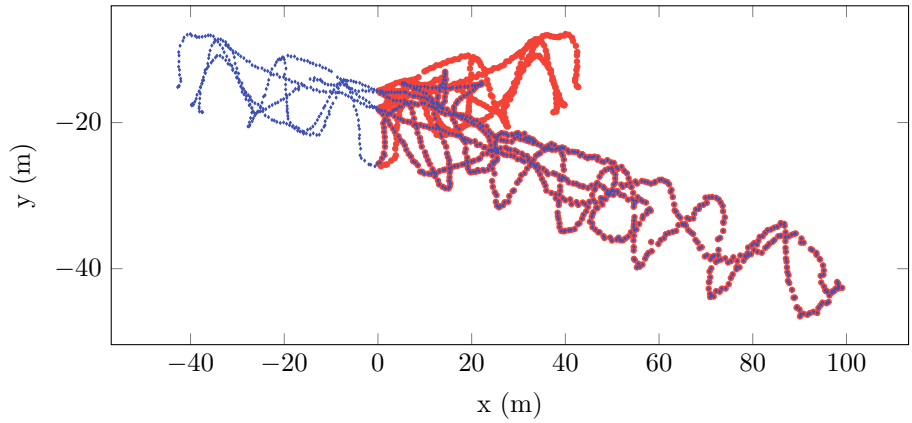
Performance for the model trained on the original data set will be compared to both the performance of the transformed data sets as to the results from the original PoseNet[Kendall et al., 2015] model. Performance is defined as average and median error for the location, orientation and half prediction. As the original PoseNet model predicts x, y, z coordinates and quaternions for the orientation their comparison is a mere indication of performance.

4.3 Results

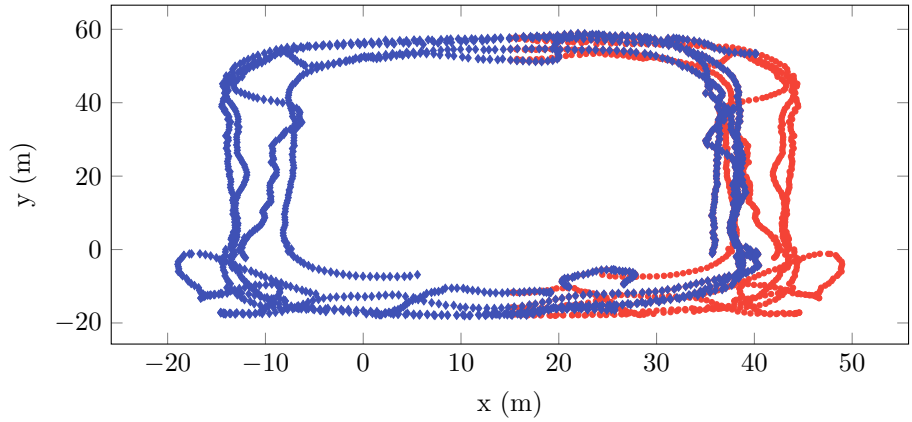
The results for the grid search on the test sets for the different data sets can be found in Table 4.1. Using these best performing parameters, the same models are tested on their validation sets whose results can be found on Table 4.2. For the Kings College data set the parameters for α, β and γ are 1 and 500 for the full representation and 0.3, 150 and 300 for the half representation. With these results, no clear distinction can be made in the error distributions visualised in Figure 4.5 while the average and median results for the location show a slight increase in performance, the orientation performance decreases.

For the St Marys Church the parameters 2 and 750 performance best on the test set for the half representation while the parameters 0.3, 150 and 300 do for the half representation. With these parameters, the visualisation of the error of each pose (Figure 4.6) shows that the full representation has a number of points that perform particularly bad while these are not present with the half representation.

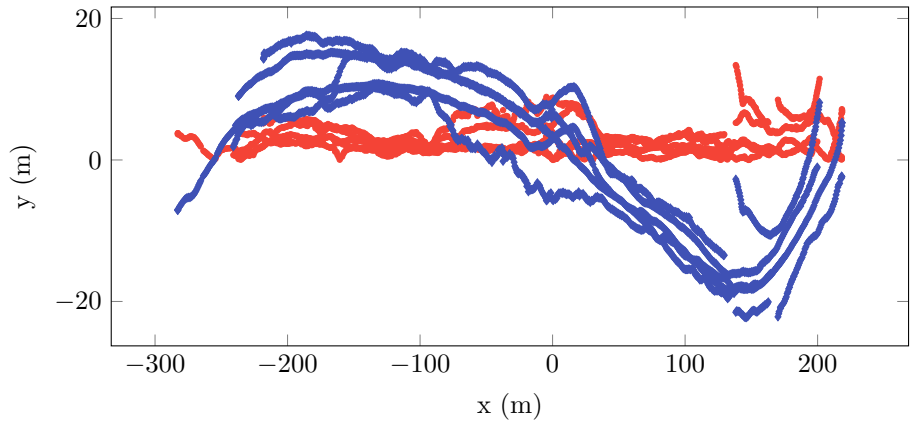
Finally, for the Street data set parameters 1, 500 and 0.3, 150 and 300 are picked for the full and half representations. Similar to the visualisation of the error for the poses of the Kings College data set, the visualisation for the Street data set, found in Figure 4.7, show no clear differences in performance.



(a) The poses from the Kings College data set before transformation (blue) and after (red).



(b) The poses from the St Marys Church data set before transformation (blue) and after (red).



(c) The poses from the Street data set before transformation (blue) and after (red).

Figure 4.4: A visualisation of what the poses look like after the symmetry is encoded in the output space of the models for the Kings College, St Marys Church and Street data set.

	α	β	γ	$\overline{p}$	$\tilde{p}$	$\overline{\theta}$	$\tilde{\theta}$	$\overline{h}$	$\tilde{h}$
Kings College									
Full									
	1	500	-	1.69	1.47	3.99	3.80	-	-
	2	750	-	1.80	1.60	4.39	3.94	-	-
	100	250	-	1.51	1.35	7.92	7.10	-	-
	250	100	-	17.62	17.01	4.29	4.51	-	-
Half									
	0.3	150	300	2.95	2.27	5.67	4.00	0.02	0.01
	100	100	250	1.52	1.29	44.19	44.39	0.52	0.52
	100	250	100	1.50	1.61	43.21	43.83	0.52	0.52
	250	100	100	1.56	1.26	44.39	44.89	0.51	0.51
	250	250	100	1.53	1.36	44.64	44.79	0.52	0.52
	250	100	250	1.99	1.65	43.75	44.07	0.52	0.52
	100	250	250	1.64	1.42	40.34	40.39	0.51	0.51
St Marys Church									
Full									
	1	500	-	4.52	3.34	6.89	5.04	-	-
	2	750	-	3.57	2.79	5.98	4.53	-	-
	100	250	-	4.01	3.56	10.58	9.43	-	-
	250	100	-	3.56	4.17	10.34	8.55	-	-
Half									
	0.3	150	300	3.41	2.96	7.46	3.82	0.08	0.02
	100	100	250	3.39	2.54	67.19	67.17	0.51	0.51
	100	250	100	3.26	2.76	67.61	65.44	0.51	0.51
	250	100	100	3.24	2.60	67.44	65.07	0.51	0.51
	250	250	100	2.96	2.45	67.43	64.65	0.51	0.51
	250	100	250	3.04	2.32	68.19	68.45	0.51	0.51
	100	250	250	3.91	3.22	67.22	66.08	0.51	0.52
Street									
Full									
	1	500	-	126.08	123.46	57.08	55.23	-	-
	2	750	-	174.36	166.28	57.98	55.39	-	-
	100	250	-	150.77	143.24	58.09	56.24	-	-
	250	100	-	142.63	143.19	59.03	62.65	-	-
Half									
	0.3	150	300	46.50	31.72	39.01	29.92	0.42	0.39
	100	100	250	203.82	210.26	75.71	81.78	0.50	0.50
	100	250	100	161.53	151.51	74.22	74.88	0.50	0.50
	250	100	100	205.38	215.40	77.64	83.13	0.50	0.50
	250	250	100	128.03	120.50	73.41	72.77	0.50	0.50
	250	100	250	66.39	51.09	69.98	70.00	0.51	0.51
	100	250	250	168.91	176.19	75.97	78.95	0.50	0.50

Table 4.1: Average and median errors in metres for the location, degrees for the orientation and half to determine the values for α, β and γ.

	α	β	γ	$\overline{p}$	$\tilde{p}$	$\overline{\theta}$	$\tilde{\theta}$	$\overline{h}$	$\tilde{h}$
Kings College									
Full	1	500	-	3.03	1.98	5.43	4.29	-	-
Half	0.3	150	300	2.95	2.27	5.67	4.00	0.02	0.01
PoseNet	-	-	-	-	1.92	-	2.70	-	-
St Marys Church									
Full	2	750	-	4.04	4.88	10.48	7.63	-	-
Half	0.3	150	300	3.41	2.96	7.46	3.82	0.08	0.02
PoseNet	-	-	-	-	2.65	-	4.24	-	-
Street									
Full	1	500	-	88.09	77.49	87.43	73.05	-	-
Half	0.3	150	300	46.50	31.72	39.01	29.92	0.42	0.39
PoseNet	-	-	-	-	3.67	-	3.25	-	-

Table 4.2: Average and median errors in metres for the location, degrees for the orientation and half for the point symmetry to create a half and full representation.

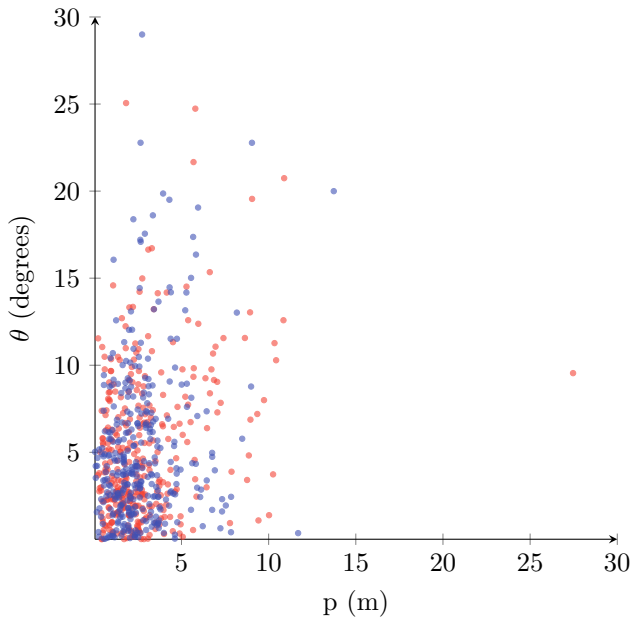


Figure 4.5: Errors for the location (p) and orientation (θ) for the full (red) and half (blue) representations for the Kings College data set.

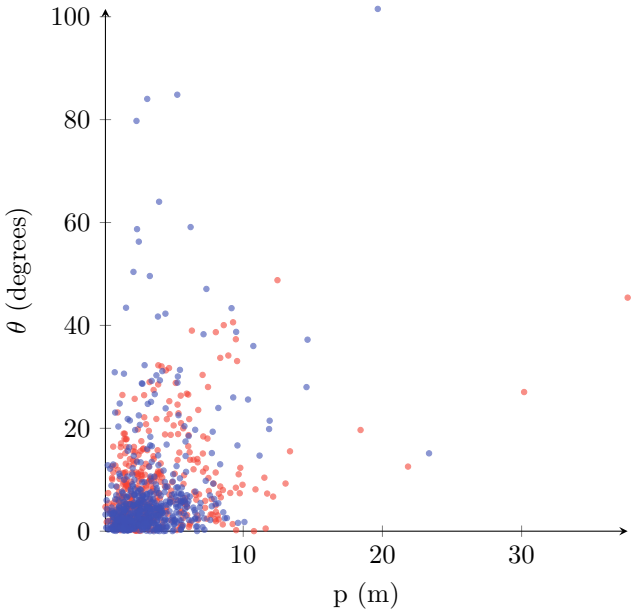


Figure 4.6: Errors for the location (p) and orientation (θ) for the full (red) and half (blue) representations for the St. Marys Church data set.

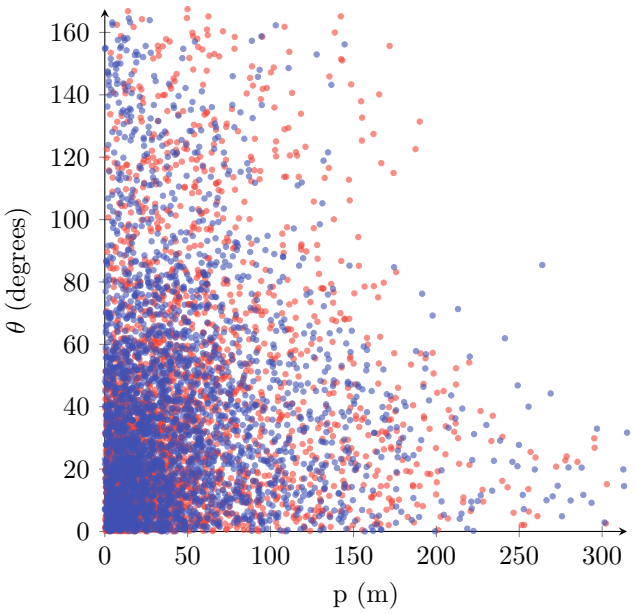


Figure 4.7: Errors for the location (p) and orientation (θ) for the full (red) and half (blue) representations for the Street data set.

4.4 Conclusions

From the results in Table 4.2 the following can be concluded: for the Kings College, St Marys Church and Street data sets, the original PoseNet model performs better than the full representation tested in this experiment. One possible cause could be the different representations used in this experiment as in our model only the x and y coordinates are predicted whereas in the PoseNet model the z coordinate is predicted as well. The orientation is predicted differently as well: our model predicts the two coordinates on a unit circle to get the orientation around a single axis, whereas PoseNet predicts the four components of a quaternion. A more fine tuned grid search to determine the parameters for the multivariate loss function might be another reason. The reason behind this experiment was however not to prove that better hyperparameters, such as the ones for the multivariate loss function, results in a better performance but to compare the influence of encoding the symmetry in the half representation.

The first thing that stands out is that the results for the Street data set are far worse than the results obtained by the original authors of PoseNet. Even with the original architecture and hyperparameters, different results were obtained. Due to this difference, the results are not comparable to the original. When observing the results for the full and half representations, the errors in prediction of the half (0.42 on average) is highly unreliable along with the high errors for the location (46.5 metres) and orientation (39.01) the created model is unusable. This is however, not against the expectations as a symmetry was forced upon the model while none was present.

The models that were created with the half and full representations for the Kings College data sets, show no significant difference for the location (1.98 and 2.27) with a P score of 0.8331 and the orientation (4.29 and 4.00) with a P score of 0.0174. The half prediction that comes with the half representation does perform well with a median error of 0.01 meaning that the model is clearly able to distinguish on which side of the gate the image is taken. The model trained on the St. Marys Church shows significant difference for the full and half representations when predicting the location (4.88 and 2.96) with a P score of 0.0007 while no significant difference can be observed for the orientation (7.63 and 3.82) with a P score of 0.65. Again, the model is also able to accurately determine the side of the church with a median error of 0.02. With a output space that is more complex when compared to the wall from Kings College, it appears that encoding the symmetry leads to higher accuracy.

When observing the visualisation of the Kings College, St. Marys Church and Street error distributions for the full and half representation, all data set appear to be similar. No clear correlation between the error for the location and orientation can be found meaning that some times the model is able to predict the orientation accurately while predicting the location wrongly or the other way around. From this we can conclude that some images do contain information from which the location or orientation can be determined, while getting the other prediction wrong.

Overall, the results indicate that encoding the symmetry in the output space of the model increases the performance of the prediction of the location with the St. Marys Church data set while performance does not decrease on any of the other predictions.

Chapter 5

Conclusions and Future Work

5.1 Conclusions and Discussion

From the experiments performed in this thesis the most valuable discovery is that when a symmetry is present, such as in the soccer field and the St. Marys Church data sets, encoding the symmetry in the output space of the prediction of the relocalisation model can increase the performance. While with the soccer task the prediction of the orientation of the robot on the field became far worse, the real world data from the Street scenes did not suffer from this side effect. For the Kings College data set, the output space might not have been complex enough, leaving no room for improvement by exploiting the symmetry. On the other hand, the Street data set had no symmetry to begin with, leading to no significant improvement, showing that it is not only a reduction in the limits of the output space that yield better results. Concluding that applying the symmetry when applicable, leads to an improvement while trying to relocalise.

A point of discussion is the decrease in performance when predicting the orientation of the robot on a soccer field with the half and quarter representations. The same error does however not appear in the prediction of the orientation for the Street scenes suggesting that the error is more likely related to the data set and less likely to the model and methods used throughout this thesis. In addition to the comparison of the full and half representation in the Street scenes data set, an additional comparison of the line and point symmetries would have been beneficial to the conclusions in this thesis. Unfortunately yet not surprisingly, no data set of real world data with known poses could be found of a location with a point symmetry. When such data set would have been available, the performance of the point and line symmetries could have been compared as well on real world data.

Finally, the lack of real world labelled soccer data results in the absence of some insightful comparisons between real world data based models and virtual data based ones even though previous work has showed that this can be successful. It would also have been interesting to first train models on virtual data, followed by training on real world data also known as transfer learning. The acquisition of real world data is therefore also one of the subjects in the next Section about the future work.

With the data available, several experiment have been conducted to explore the possibilities of the use of symmetries on deep convolutional neural networks. The results from the experiments have showed that the use of the symmetry can be beneficial when determining the pose.

5.2 Future Work

As is typically the case, during the research period of this thesis several extensions have been thought of yet not explored due to various reason of which the time constraint is the most important one. Two major topics are of our interest: towards mobile robots and end to end learning of symmetries without manual explicit encoding in the output space of the model.

5.2.1 Mobile Robots

At the start of this thesis, the first idea was to develop a relocalisation method that could be run on a Nao robot, this turned out to be difficult because of two reason: no data set exists of images from the robot with known poses and it turns out that creating one is a non-trivial task. The second reason is that performing inference of a convolutional neural network on the robot itself requires strict limitations on the architecture.

Real World Data

As explained in Section 1.2.1, trying to create a data set of an actual robot walking around a full sized field, which is nine by six metres, proved to be beyond the capabilities of the tracking system available to us. To create a model that is able to predict the pose of images from the actual robot, realistic images for both training and testing captured by the robot are however a necessity. Apart from attempting to use the tracking system to create labels for the images, two other methods can be tried for future research. The first uses the structure from motion algorithm from Section 2.1.2 where from two subsequent images, the movement of the robot is estimated. As the error in the determined global position of the robot is the sum of all previous movements of subsequent images, the error in the global position accumulates over time. Nonetheless, this is a popular method to create poses for images and might be good enough to provide insight into the capabilities of the used model architecture on a real world data set.

The second method would be to create a more realistic simulation that would produce images that are nearly indistinguishable from real, like the KITTI data set described in Section 1.2.3. By creating such a simulation, the creation of new data sets with certain features such as walking along a line, placement of other robots or movement of the ball, would become far easier than trying to create these with a real robot. This simulation could then also be used for various other vision related tasks such as recognising other robots, the goal or the ball as perfect ground truth can be generated without the need for manual annotation. To validate that this simulation is realistic, images from the real world robot would however still be required.

Reduce Computational Requirements

The other major subject of interest relates to being able to perform inference on the robot itself. While inference time on a GPU enabled machine is reasonable for real time usage, the Nao robot that has to be used in the soccer matches only has a single core CPU from 2004. In this thesis the GoogLeNet architecture has been used to extract features from the images to predict poses. It might be the case that this architecture provides more detailed features than strictly necessary to relocalise. Exploring which layers in the GoogLeNet architecture provide useful features while disabling the others could result in a speed up of the inference time enabling the robot to perform inference in reasonable time. Trying out networks with faster inference time altogether that require less computational operations, while maintaining the same accuracy or even improve it is another research direction. This method is supported by the preliminary results found in Table 5.1 which are obtained by initialising with a model trained on the ImageNet data set and further trained with the St. Marys Church data set, and adapting each model architecture to include the regression layers to be able to predict the pose.

These results indicates that the AlexNet [Krizhevsky et al., 2012] network is able to achieve similar results while being faster to train and to perform inference [Canziani et al., 2016]. SqueezeNet [Iandola et al., 2016] has approximately 1.2 million parameters compared to GoogLeNets approximately 11 million parameters and is also able to perform localisation with the full representation. For the half representation, the accuracy decreases which might be due to the hyperparameters of the multivariate loss function. These results indicate that other faster networks are also able to create usable features for relocalisation. Creating a network specifically designed for the robot might therefore also be able to create usable features enabling to use the proposed methods of this this in real time, on the robot. For the robot network one topic of interest is the use of binary networks [Rastegari et al., 2016] [Wu et al., 2016] that reduce the precision of network layers to be able to use bit wise operations to parallelise computations in both the convolutional and fully connected layers.

Architecture	$\overline{p}$	$\tilde{p}$	$\overline{\theta}$	$\tilde{\theta}$	$\overline{h}$	$\tilde{h}$
GoogLeNet						
Full	3.03	1.98	5.43	4.29	-	-
Half	3.40	2.86	7.57	4.06	0.09	0.03
AlexNet						
Full	6.86	5.05	13.66	6.10	-	-
Half	5.59	4.52	12.90	4.85	0.17	0.06
SqueezeNet						
Full	7.02	6.50	9.46	6.15	-	-
Half	17.75	17.34	10.58	5.54	0.09	0.01

Table 5.1: Average ($\overline{p}$, $\overline{\theta}$ and ($\overline{h}$) and median ($\tilde{p}$, $\tilde{\theta}$ and $\tilde{h}$) errors for the location, orientation and half predictions for the previously used GoogLeNet, AlexNet and SqueezeNet. Preliminary results indicating that other, faster, network architectures such as AlexNet are also able to perform the relocalisation task. SqueezeNet performs worse with the half representation while performing similar to the other architectures on the full representation.

5.2.2 End to End Learning of Symmetry Exploitation

In the experiments described in this thesis, for both the soccer and the street scenes data sets, the symmetry was enforced manually by encoding it in the predictions of the used network. In future work, ideally, these symmetries would be detected by the network itself meaning that knowledge about similar inputs resulting in different outputs needs to be captured. To do this, the architecture of the network needs to implement the transformation of the coordinates itself. As an initial experiment we suggest adding a line symmetry layer where the transformed pose (p_h), orientation (θ_h, θ_h) and half (h) are formally defined as follows:

$$h = \left(p - b \begin{bmatrix}0\\1\end{bmatrix}\right) \times \mu > 0 \tag{5.1}$$

Where h is therefore a binary value: either one or zero. As no ground truth is available for this variable it is a latent variable and used to determine the location and orientation:

$$p_h = p - 2h \frac{\left(p - b \begin{bmatrix}0\\1\end{bmatrix}\right) \cdot \mu_\perp}{\mu_\perp \cdot \mu_\perp} \cdot \mu_\perp \tag{5.2}$$

$$\theta_h = -h\left(\theta + 2(-\theta \cdot \mu)\mu\right) + (1-h)\theta \tag{5.3}$$

Where μ depicts the direction of the symmetry line (and $\mu_\perp$ the perpendicular direction) and b the translation of the symmetry line from the origin thus introducing two variables that encode the symmetry. These two parameters can be learned as is typical in convolutional neural networks by using the gradients of μ and b. How much influence the addition of these two parameters to the millions of parameters already in the convolutional neural network would have to be determined experimentally.

This implements an explicit line symmetry layer, which could be seen as encoding prior knowledge in the network architecture, yet without explicitly giving the specific symmetry to the network. As the method is optimised during training, the model might use a different idea of symmetry then we would. As this method does not encode that images on both sides of the symmetry need to be similar per the definition of a symmetry, it is also possible that the model optimises μ and b in another way. Just like the experiments done in this thesis, the symmetry applies to the entire output space of the data set while symmetries that only apply to a part of the output space might be of interest as well.

In this thesis we have explored using given symmetries and experimentally showed that this could be beneficial when trying to relocalise on a virtual soccer field or a real world street scene.

Bibliography

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417.

[Canziani et al., 2016] Canziani, A., Paszke, A., and Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

[Cao et al., 2017] Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*.

[Cassinis et al., 2002] Cassinis, R., Duina, D., Inelli, S., and Rizzi, A. (2002). Unsupervised matching of visual landmarks for robotic homing using fourier–mellin transform. *Robotics and Autonomous Systems*, 40(2):131–138.

[Cicconet et al., 2017] Cicconet, M., Birodkar, V., Lund, M., Werman, M., and Geiger, D. (2017). A convolutional approach to reflection symmetry. *Pattern Recognition Letters*, 95:44–50.

[Collett et al., 1992] Collett, T. S., Dillmann, E., Giger, A., and Wehner, R. (1992). Visual landmarks and route following in desert ants. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 170(4):435–442.

[Cooley and Tukey, 1965] Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301.

[Cula and Dana, 2001] Cula, O. G. and Dana, K. J. (2001). Compact representation of bidirectional texture functions. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE.

[Davison et al., 2007] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067.

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

[Deng et al., 2011] Deng, J., Satheesh, S., Berg, A. C., and Li, F. (2011). Fast and balanced: Efficient label tree learning for large scale object recognition. In *Advances in Neural Information Processing Systems*, pages 567–575.

[Dosovitskiy et al., 2015] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766.

[Duda and Hart, 1972] Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

[Engelson and McDermott, 1992] Engelson, S. P. and McDermott, D. V. (1992). Error correction in mobile robot map learning. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2555–2560. IEEE.

[Fürtig et al., 2017] Fürtig, D.-I. A., Brast, J. C., Ditzel, S., Hammer, H.-J., Hess, T., Rinfreschi, K., Siegl, J.-M., Steiner, S., Weiglhofer, F., and Wörner, P. (2017). Team description for robocup 2017. In *RoboCup 2017: Robot World Cup XXI*.

[Gaidon et al., 2016] Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. *CoRR*, abs/1605.06457.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK.

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

[Iandola et al., 2016] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

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

[Julier and Uhlmann, 1997] Julier, S. J. and Uhlmann, J. K. (1997). A new extension of the kalman filter to nonlinear systems. In *Int. symp. aerospace/defense sensing, simul. and controls*, volume 3, pages 182–193. Orlando, FL.

[Kalman et al., 1960] Kalman, R. E. et al. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

[Kendall et al., 2015] Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946.

[Koenderink and Van Doorn, 1991] Koenderink, J. J. and Van Doorn, A. J. (1991). Affine structure from motion. *JOSA A*, 8(2):377–385.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Labbe and Michaud, 2014] Labbe, M. and Michaud, F. (2014). Online global loop closure detection for large-scale multi-session graph-based slam. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2661–2666. IEEE.

[Laskar et al., 2017] Laskar, Z., Melekhov, I., Kalia, S., and Kannala, J. (2017). Camera relocalization by computing pairwise relative poses using convolutional neural network. *arXiv preprint arXiv:1707.09733*.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[Liang et al., 2013] Liang, J. Z., Corso, N., Turner, E., and Zakhor, A. (2013). Image based localization in indoor environments. In *Computing for Geospatial Research and Application (COM. Geo), 2013 Fourth International Conference on*, pages 70–75. IEEE.

[Liu et al., 2013] Liu, J., Slota, G., Zheng, G., Wu, Z., Park, M., Lee, S., Rauschert, I., and Liu, Y. (2013). Symmetry detection from realworld images competition 2013: Summary and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[Loy and Eklundh, 2006] Loy, G. and Eklundh, J.-O. (2006). Detecting symmetry and symmetric constellations of features. In *European Conference on Computer Vision*, pages 508–521. Springer.

[Menegatti et al., 2003] Menegatti, E., Zoccarato, M., Pagello, E., and Ishiguro, H. (2003). Hierarchical image-based localisation for mobile robots with monte-carlo localisation. In *Proc. of European Conference on Mobile Robots (ECMR'03)*, pages 13–20.

[Newman and Ho, 2005] Newman, P. and Ho, K. (2005). Slam-loop closing with visually salient features. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 635–642. IEEE.

[Nistér et al., 2004] Nistér, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. Ieee.

[Norouzi et al., 2013] Norouzi, M., Mikolov, T., Bengio, S., Singer, Y., Shlens, J., Frome, A., Corrado, G. S., and Dean, J. (2013). Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*.

[Patraucean et al., 2013] Patraucean, V., Grompone von Gioi, R., and Ovsjanikov, M. (2013). Detection of mirror-symmetric image patches. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[Rastegari et al., 2016] Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer.

[RoboCup Technical Committee, 2017] RoboCup Technical Committee (2017). Robocup standard platform league (nao) rule book. `http://spl.robocup.org/wp-content/uploads/downloads/Rules2017.pdf`.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE.

[Smeulders et al., 2014] Smeulders, A. W., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., and Shah, M. (2014). Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468.

[Smith and Cheeseman, 1986] Smith, R. C. and Cheeseman, P. (1986). On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[T. Hess and Ramesh, 2017] T. Hess, M. Mundt, T. W. and Ramesh, V. (2017). Large-scale stochastic scene generation and semantic annotation for deep convolutional, neural network training in the robocup spl. In *RoboCup 2017: Robot World Cup XXI, LNAI*.

[Tao et al., 2016] Tao, R., Gavves, E., and Smeulders, A. W. (2016). Siamese instance search for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1420–1429.

[Torralba et al., 2008] Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970.

[Wei et al., 2016] Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. (2016). Convolutional pose machines. In *CVPR*.

[Westoby et al., 2012] Westoby, M., Brasington, J., Glasser, N., Hambrey, M., and Reynolds, J. (2012). 'structure-from-motion'photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314.

[Wu et al., 2011] Wu, C. et al. (2011). Visualsfm: A visual structure from motion system.

[Wu et al., 2016] Wu, J., Leng, C., Wang, Y., Hu, Q., and Cheng, J. (2016). Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828.