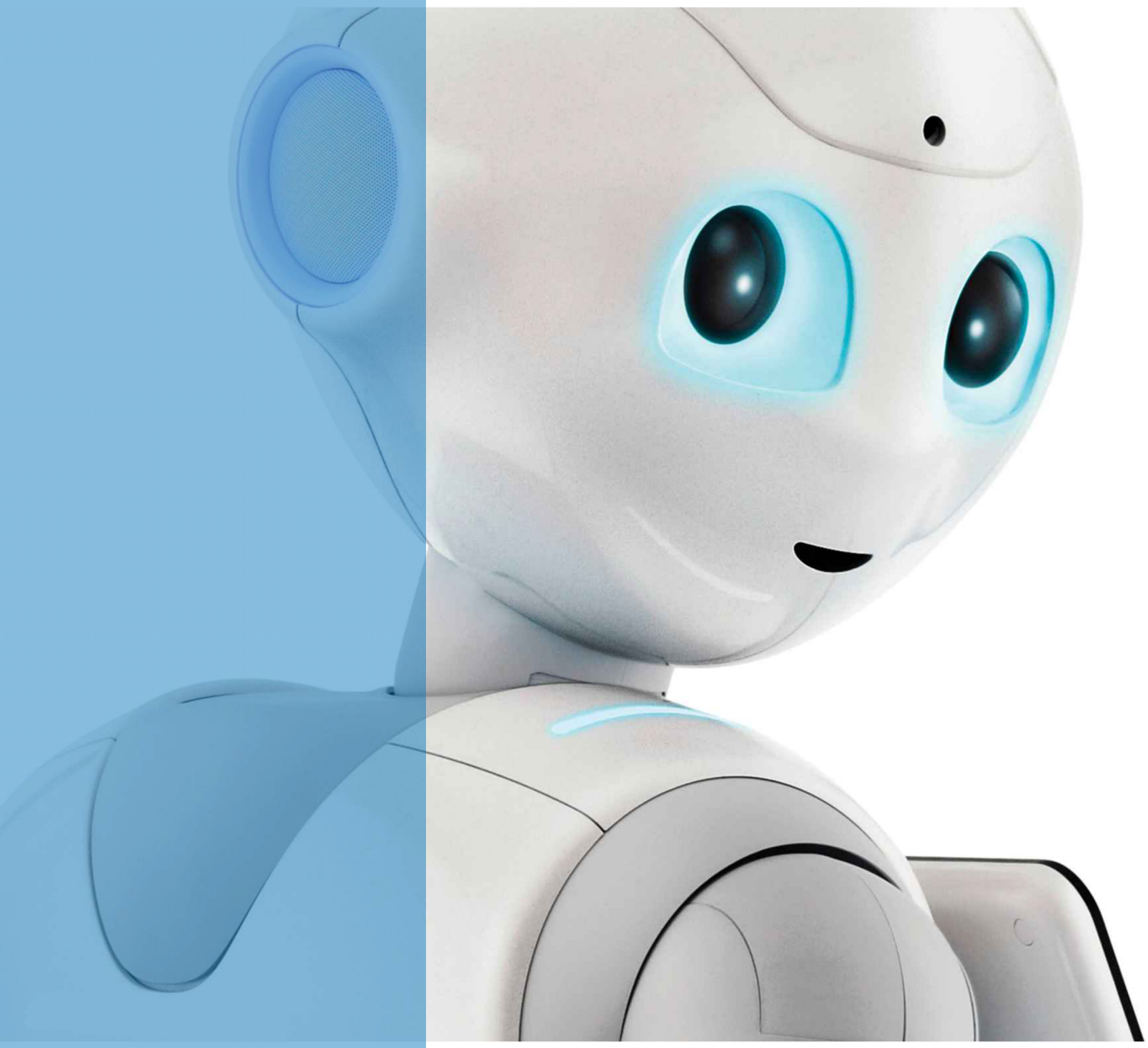
People detection on the Pepper robot using Convolutional Neural Networks and 3D blob detection



Jonathan Gerbscheid
University of Amsterdam
Faculty of Science
BSc Artificial Intelligence

Cover photograph courtesy of SoftBank Robotics,
https://www.softbank.jp/robot/special/pepper/.

People detection on the Pepper Robot using Convolutional Neural Networks and 3D Blob detection.

Jonathan R. Gerbscheid
9123456

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
dr. A. Visser

Intelligent Robotics Lab
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

July 2th, 2017

Contents

1 Introduction **2**

2 Platform **3**
2.1 2D cameras 3
2.2 3D sensor 4
2.3 Software 6

3 Theoretical Foundations **6**
3.1 Convolutional Neural Networks 6
3.2 Inception architecture 7

4 Related Work **9**

5 Methodology **10**
5.1 Detection using a Convolutional Neural Network 10
5.1.1 Datasets 11
5.1.2 Parameter Optimization 11
5.1.3 Training 12
5.1.4 Filtering Results 12
5.2 Detection using Flood Fill on a 3D Point Cloud 13
5.2.1 Adapting Flood Fill for 3D Data 13
5.2.2 Flood Fill Candidate Selection 14
5.3 Combining the Detectors 16

6 Evaluation **18**
6.1 CNN Performance 19
6.2 Blob detector performance 19
6.3 Combined Detection Performance 20

7 Discussion **22**

8 Conclusion **22**

9 Future Work **23**

A Appendix **26**

Abstract

In this thesis, the problem of people detection on the Pepper robot is explored. People detection is a critical part of human-robot interaction and advancements in people detection will improve the level of interaction that can be achieved. It has been approached using different sensors and techniques, however the capabilities of detection with the Pepper have not yet been examined properly. Detection techniques using the different sensors available to the Pepper are explored and a state of the art convolutional neural network and 3D blob detector are developed. The detectors are then combined using a detection history based approach. Results show that performance of the CNN, although high for cases with 1-3 test subjects, decreases significantly in crowded settings. The addition of 3D data to reuse previous detections was shown to increase recall, however due to the limited range of the 3D sensor, recall remained lower than that achieved on the lower person count test cases. Additionally, future work includes the use of 3D data to separate detections made on 2D images, the use of different network architectures and the use of more complex machine learning techniques for 3D people classification.

1 Introduction

In the past Artificial Intelligence has often been associated with robotics in popular culture and literature. Nowadays the applications of AI are much more varied and it is used in many fields of research. The idea of robots assisting people in their daily lives, however, is still as strong as before. With this idea of robots assisting and servicing humans, the RoboCup @Home league[1] was founded. Since the 2017 competition, the @Home league has been split into three different leagues, each focusing on different aspects of assistance robotics: The Domestic Standard Platform League (DSPL), Social Standard Platform League (SSPL) and Open Platform League (OPL). Both the DSPL and SSPL league make use of a standard robot, the Toyota HSR and the Softbank Pepper respectively. Due to the recent introduction of these robots, the capabilities of their hardware for the various tasks in the @home competition have not yet been determined. In order to better understand what can be achieved with these robots, it is necessary to test the sensors in the context of the @Home challenge is needed. The Softbank Pepper, used for the SSPL, has a large array of sensors which can be used for a variety of tasks, see section 2, one of these tasks is people detection. People detection is the problem of finding humans in the robots surroundings, this is essential for @Home League[1] as it focuses on interaction between robots and humans, which would not be possible if the robot is unaware of any humans in its surroundings. People detection is especially important to all SSPL challenges [1], as the robot must actively seek contact with people, which requires the detection to be even more stable and accurate. The different sensors of the Pepper allow it to take a multisensor approach to human detection. This research will explore how people detection can be solved on the Softbank Pepper using state of the art image recognition techniques on 2D and 3D images. More formally, the aim of this project is thus to:

> ***Develop*** *a People detection method using the different sensors available to the Softbank Pepper and find the best way of* ***combining*** *the different detection methods.*

In order to achieve the research goal, the following questions should be answered:
Q1 How can the different sensors of the Softbank Pepper be used to detect people?
Which can be split into the subquestions:

1. What sensors are relevant for people detection?
2. Which detection method is suitable for each sensor?
3. How can the different sensors be combined?

And secondly:
Q2 Does the inclusion of additional detection methods improve the overall performance in people detection?
With the subquestions:

1. What is the performance of the 2D image convolutional neural network?

[1] `http://www.robocupathome.org`

2. How much does the addition of a 3D people detector improve the baseline people detection performance?

The Pepper comes equipped with two 2D camera's, a 3D camera, 6 Laser range finders and infra-red and sonar obstacle detectors [2]. A baseline 2D people detection method using convolutional neural networks is implemented, this network is trained on a dataset consisting of various people detection datasets from TU-Dresden[2][3] because the images in these datasets are similar to images taken by the Pepper. A 3D blob detection method using the depth camera is also be developed. A new dataset containing corresponding 3D and 2D images is created and a 3D people detection algorithm based on the flood fill algorithm is developed. Both methods are combined using a geometrical translation approach and finally, the performance of the baseline 2D detection method is compared to the combined 2D and 3D approach.

2 Platform

This section covers the technical aspects of the Softbank Pepper robot that are relevant to this research. The Pepper robot is around 1.2 meters in height, see Figure 1 and weighs 29kg. It is equipped with a microphone, two 2D cameras, a 3D Sensor, Laser Range Finders, Infrared Sensors and two ultrasonic sensors[2]. Is this section we will look at the 2D cameras and the 3D sensor in more detail.

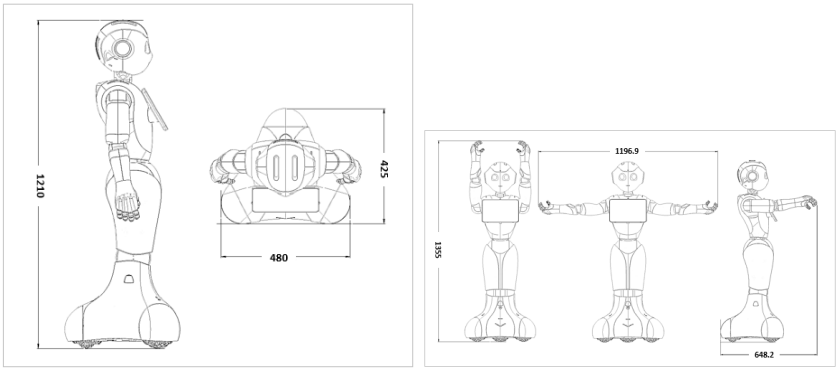


Figure 1: Dimensions of the Pepper in mm[2]

2.1 2D cameras

The Peppers two cameras are located on the forehead and in the mouth of the robot, Figure 2. Both cameras have a horizontal field of view of 55.2°and a vertical field of view of 44.3 °, the fields of view of the two cameras intersect from ~100 cm. When Pepper is looking straight ahead, the lower camera will often only see the floor. However because humans are generally much taller, it will generally be facing up when interacting directly with humans.

[2] http://doc.aldebaran.com/2-4/family/pepper_technical/index_pep.html

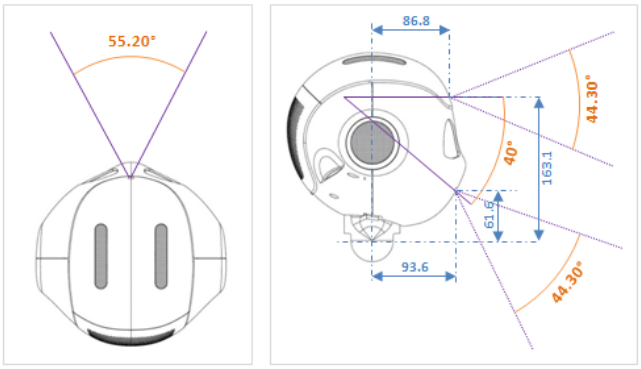


Figure 2: 2D cameras located on the head of the Pepper[2]

2.2 3D sensor

The 3D Sensor used in the Pepper is a version of the Asus Xtion 3D sensor and is located behind the eyes of the Pepper. Its horizontal and vertical field of view is slightly larger than that of the 2D cameras and it is pointed in the same direction as the upper 2D camera, Figure 3. It reports distances to points in the range of 40 to 400cm in a number of different formats. The Sensor is often unable to find the range to a certain point when the surface reflects the laser, this results in data that, while generally stable, contains small gaps. While the sensor should report information up to 4 meters, the limited resolution of the camera results in noisy data. On visual inspection of the data, humans are clearly defined up to 3 meters and a blob remains visible up to the maximum range. Depth information is gathered by the 3D sensor in the Pepper through the use of projected light patterns. In this method a pattern of narrow bands of near-infrared light is projected onto the surroundings, this pattern, while being perceived as a straight lines from the projecting camera, is seen distorted through the second camera, see Figure 4. These lines can be used for a geometrical construction of the shape of the surface.

This way of obtaining depth information is quick and costs little processing power, the downside however is that it is not possible to use the sensors in an outdoor setting or any setting where near-infrared light is produced, due to the inference of natural light with the pattern recognition.

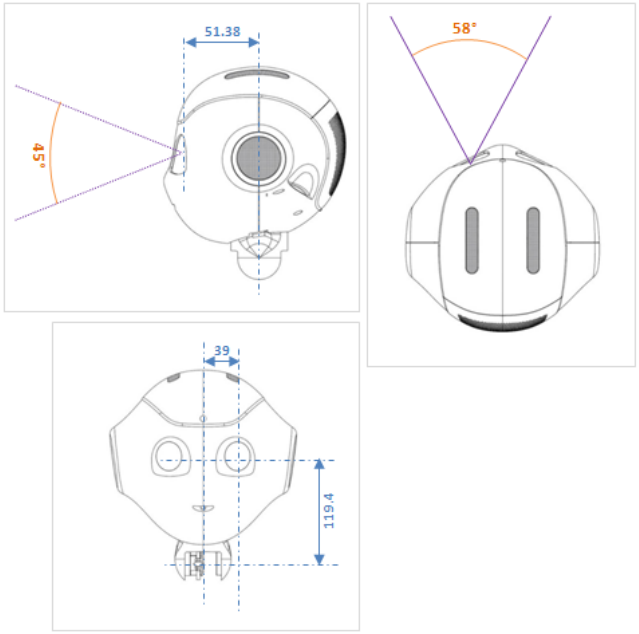


Figure 3: 3D camera located in the left eye of the Pepper[2]

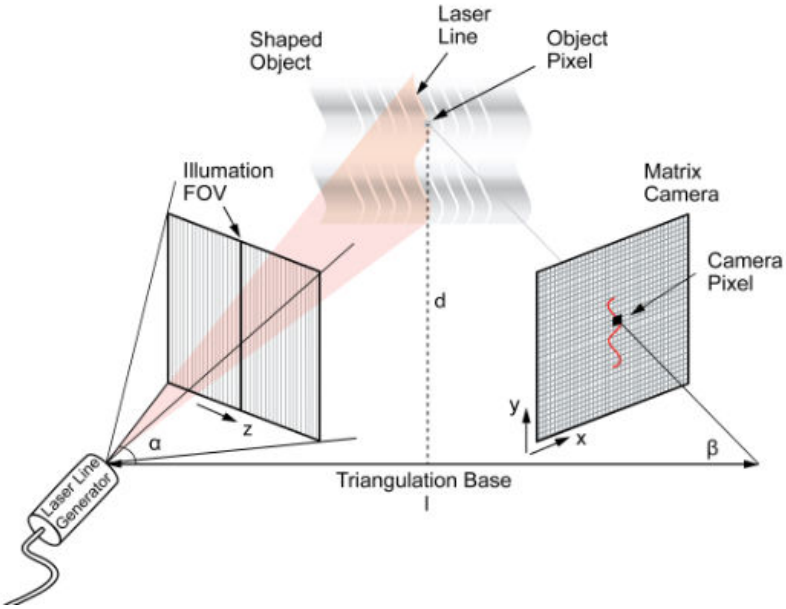


Figure 4: Structured light scanning, a straight line pattern is projected onto a curved surface[3].

2.3 Software

A general note for this project is that all prediction and utility code is written in Python 2.7 in order to be compatible with the naoqi 2.5 operating system used on the Pepper robot. A full overview of all packages used can be found in Section section A.

3 Theoretical Foundations

3.1 Convolutional Neural Networks

Convolutional neural networks first gained popularity when the AlexNet outperformed all other approaches to image classification in the ImageNet challenge [4]. Since then they have become the most used approach to almost all vision related problems[5]. Convolutional neural networks are a variation on feed-forward neural networks in which the general idea is to capture shapes at different levels of abstraction and combine them to find complex patterns, such as humans. CNNs consists of convolutional, pooling or sub sampling, normalization and fully connected layers, connected to each other sequentially. The first operation in a CNN is the convolutional operation, from which the name of the network is derived. In this operation, a filter representing a certain feature is moved across the input image, resulting in a score map for every position, Figure 5, which can also be interpreted as an image again.

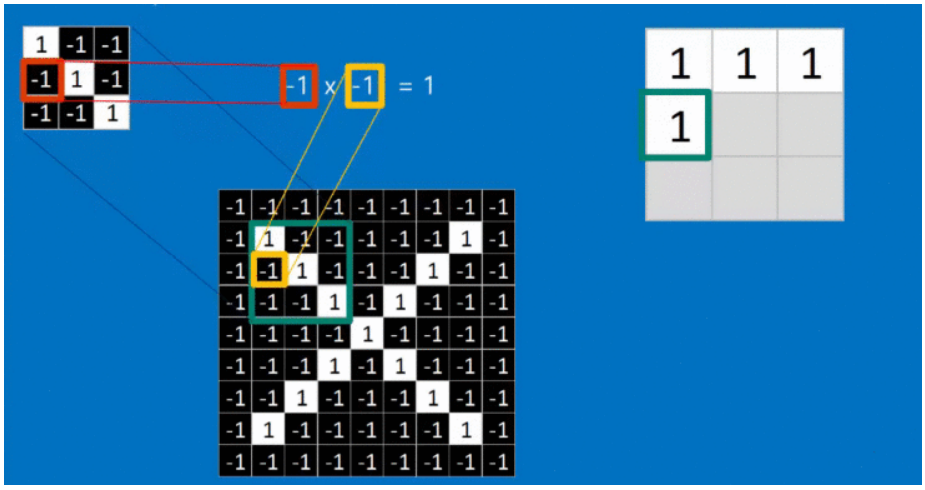


Figure 5: Applying a 3x3 filter on an 9x9 image, generating a new 3x3 image[4].

This score map is then rectified, this is usually through the use of the ReLU function. The ReLU function, Equation 1, sets every negative number to 0 and functions as a rectification and has recently been used over the sigmoid function[6], as it has been shown to increase performance.

$$ReLU(x) = max(0, x) \tag{1}$$

[3]source: `http://www.thefullwiki.org/Structured-light_3D_scanner`

The next step is the Pooling or sub sampling. In this step the image or score map received is downsized by sliding a window over the image and taking the maximum value in the window, this value is then used as in the new smaller image. The distance with which the window moves is called the stride of the window, the size and stride of the chosen window determine the size of the new image that is generate. This process can be seen in Figure 6.

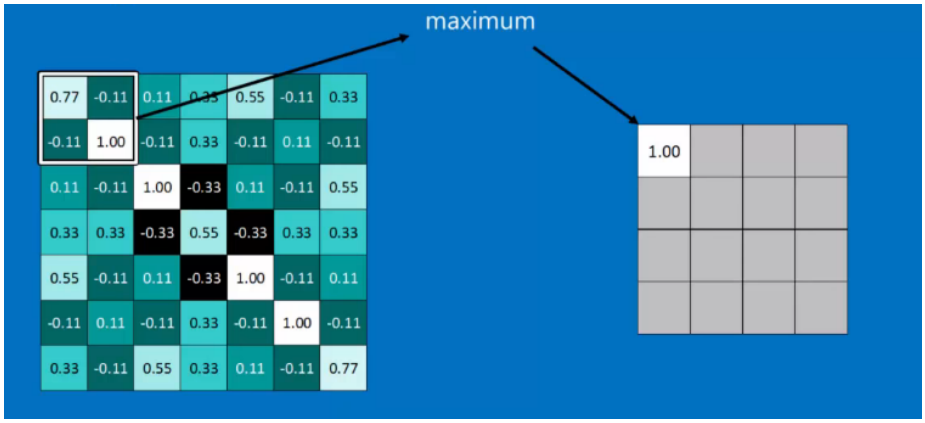


Figure 6: 4x4 Max Pooling operation.
Left: original image, Right: resulting image[4]

The last type of layer is the fully connected layer, usually followed by a softmax activation function to normalize the results, in a fully connected layer all image values received are put in a single array and connected to a number of nodes that represent the classes. All nodes have a certain likelihood to indicate a class, so if a node that is strongly connected to a class has a high value, it increases the score of that class, see Figure 7.
These layers form the basics of all CNN's and can be used in combination with each other, sequential or parallel, to classify an image. However, before classification is possible using the above structures the parameters of this network need to be learned. The filters, which are able to learn distinctive features, applied in convolutional layers and the weights of the connections in fully connected layers are learned during the training of the network. This learning is done through the use of the backpropagation algorithm.

3.2 Inception architecture

The inception architecture used in this paper was created by Szegedy et al. [7] and improves upon the standard elements of CNNs by the addition of a new layer type called the inception layer, Figure 8, and auxiliary classification structures.

[4]source: `http://brohrer.github.io/how_convolutional_neural_networks_work.html`

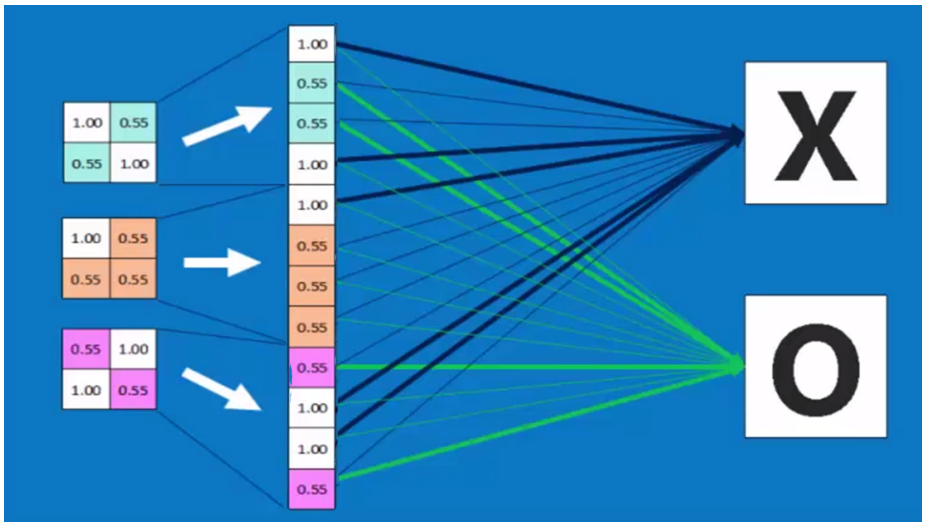


Figure 7: Fully connected layer classifying between two classes[4]

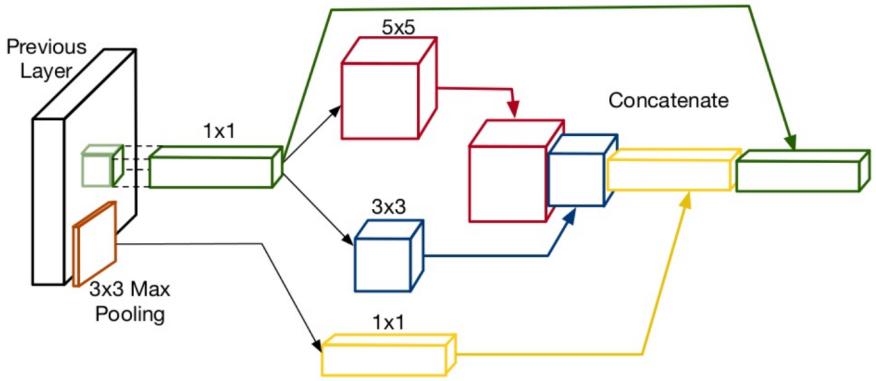


Figure 8: Inception layer[5]

The Inception layer type is comprised of several of the regular operations present in CNNs and using them in parallel. The idea behind this is to take advantage of the benefits of the different sizes of convolutions without having to chose just one. 1x1 convolutions capture information across image channels, red-green-blue channels of the image for example, and reduces dimensionality. 3x3 and 5x5 convolutions capture spatial information on different scales and max pooling layers capture additional information missed by the convolution operations. In the end the result is concatenated into a single output vector that can be used as input for the next layer. Auxiliary classifiers in the Inception architecture are smaller convolutional networks connected to Inception layers in the middle of the network. These networks are comprised of an average pooling layer followed by a 1x1 convolutional layer followed by two fully connected layers and finally

a softmax activation layer, Equation 2.

$$\sigma(z)_j = \frac{e^{z_k}}{\sum_{k=1}^{K} e^{z_k}} for j = 1, ..., K \tag{2}$$

The loss of these classifiers is weighted and added to the total loss of the network. The purpose of these layers is to provide regularization and help capture detail at different levels of abstraction, this is necessary because the depth of the network could result in detail getting lost due to the long path to the final loss function. These networks are only used during the training phase and are meant to improve performance, they are discarded for testing. For a model of the auxiliary classifiers see Figure 9.

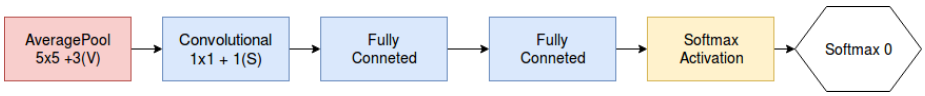


Figure 9: Auxiliary convnet classifier

The entire Inception network is then created by combining traditional convolutional layers with Inception layers. Starting with standard convolutional layers in the lower layers of the network for memory efficiency and Inception layers higher up, with max pooling layers added at specific points. The entire architecture can be seen in Figure 10.

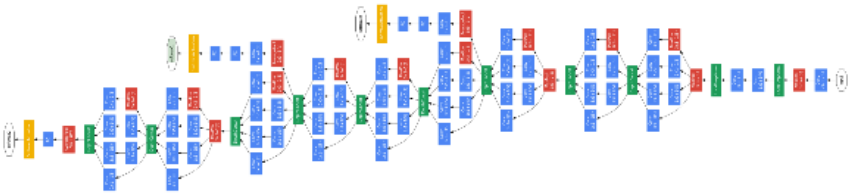


Figure 10: InceptionV1 architecture
Convolutional layers in blue, pooling layers in red, connecting layers in green and softmax activation layers in yellow [7].

4 Related Work

Several approaches to human detection have been taken in the past by teams in the @Home competition. Correa et al.[8] used a system in which different stages of detection and recognition were used to detect people through the use of both 2D visual images and thermal information. In their approach a full body detection was first performed using the thermal information, after which candidates for facial recognition are gathered by running face detection on the

[5]source: `https://www.slideshare.net/aurot/googlenet-insights`

obtained full bodies. They were able to recognize humans in a robust and stable manner, through the use of the different sensors available to them. Their method however, was unable to detect people that were not facing the robot directly due to their reliance on facial recognition.

A different approach taken by Belle et al. [9] is to use random forests for both detection and recognition. By first training a detection random forest (RF) and retraining it when faces are detected by adding additional labels, the RF is able to detect, learn and recognize in one step, while never stopping the system.

Lastly, Figueroa et al. [10] used a Microsoft Kinect in combination with color images to detect objects using SIFT to generate interest points. These interest points are then used to generate a region of interest, which is transposed on the depth image. Finally the centroid of this region of interest is used to determine the distance to the object. This approach is similar to the one taken in this paper, however this paper focuses on stability through a detection history and people detection rather than general object detection.

5 Methodology

To recall, the aim of this research is to:

> ***Develop*** *a people detection method using the different sensors available to the Softbank Pepper and find the best way of* ***combining*** *the different detection methods.*

The Methodology consists of three Sections, Section subsection 5.1 details what base network was used, what dataset the network was trained on, what parameters were adjusted to make the network suitable for people detection. Section subsection 5.2 describes the details of the stereo-image people detection method starting with the adaptation of the flood fill algorithm[11] to stereo images and consequently how the adapted algorithm is used to generate candidate blobs for people detection. finally, Section subsection 5.3 explains how the two detection methods are combined to provide the final predictions. People detection is usually implemented as either face detection or full-body detection depending on the specific problem[12]. Stewart et al. [13] used head detection to find people in crowded scenes such as bars, while using full body detection to detect pedestrians on a street. As the aim of this research is to combine different sensors, the choice was made to use full-body detection as the level of detail provided by the 3D sensor, while high enough for tasks such as full body recognition, is not accurate enough for tasks such as head detection at the range required for this research [14].

5.1 Detection using a Convolutional Neural Network

Because of the time restraints on this project and the high performance of existing implementations the choice was made to use the Tensorbox object detection framework[6] and retrain the supplied networks for people detection. The Ten-

[6]https://github.com/TensorBox/TensorBox

sorbox framework allows the use of either the Inception_v1 [7] or Resnet [15] classification networks combined with an attention network to identify potential objects. For this research the choice was made to use the Inception_v1 network because tests showed no significant difference between the two on the data used for this research and the Inception_v1 network has a shorter training duration.The networks are implemented in Tensorflow [16] and all testing/training code is written in Python.

5.1.1 Datasets

There are several datasets for full body people detection available, multiple datasets will be used in order to find the best performing dataset for people detection on the Pepper robot. The first dataset that will be used is the TUD-Brussels[2] dataset. This dataset consists of 1018 images taken from a car driving through a busy city centre. The second dataset that will be used is the TUD-Motionpairs dataset, this dataset contains 2184 images taken at eye level from a stationary camera at various public squares and shopping streets. Example images with bounding boxes from these datasets can be seen in Figure 11. Finally the datasets will be combined into a single larger dataset, this final dataset also contains images from the smaller TUD-crossing, TUD-campus and TUD-pedestrian datasets, which are similar to the TUD-Motionpairs dataset. All images together add up to a dataset of 4130 images. An additional note on this dataset is the incompleteness of the annotations. It is often the case in the annotation of networks that the choice to label a certain object that is very obscured or distant is not straightforward, however several images in this dataset miss annotations that should decidedly be present. It seems however that this does not impact the learning of distinctive features significantly, as these examples are generally still detected during training.

Figure 11: Dataset examples. Left: TUD-Brussels, Right: TUD-Motionpairs

5.1.2 Parameter Optimization

Tensorbox allows for the customization of a number of parameters of the network. The first parameter that was modified was the size of the input image and with that the size of the grid used by the network. The original size of the input image is 480 by 640, but a version of the network for images with a size of 240 by 300. The grid sizes of these networks can be seen in Table 1.

Table 1: Sizes for the different networks used

	image width	image height	grid width	grid height	region size
max size	480	640	15	20	32
small size	240	300	10	8	30

The use of a smaller of network speeds up the training and evaluation of the network. However, there is a slight performance decrease, see Figure 12, which is unnecessary as this research is not concerned with optimizing the runtime of the network.

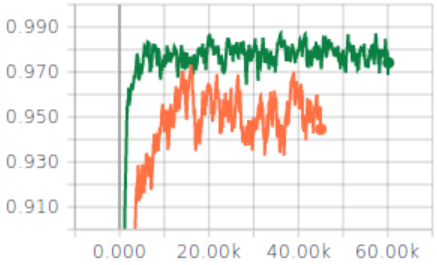


Figure 12: Test accuracies of the network trained on full size images in green and the network trained on half size images in orange

5.1.3 Training

As has been mentioned in subsection 5.1, both the Resnet and Inception_V1 networks are available in Tensorbox. Both networks were trained until their weights, accuracy and loss functions stabilized, this takes around 40000 iterations of training. Training was done on a GTX960 GPU and takes around an hour to complete. For graphs of the accuracy during the training and testing of the network on the combined dataset, see Figure 13.

5.1.4 Filtering Results

The default way to use tensorbox uses a threshold on the confidence scores to determine final detections. The default value for this threshold is 0.5, however testing showed that the confidence could be lowered to 0.1 in order to increase recall, without having an impact on the precision.

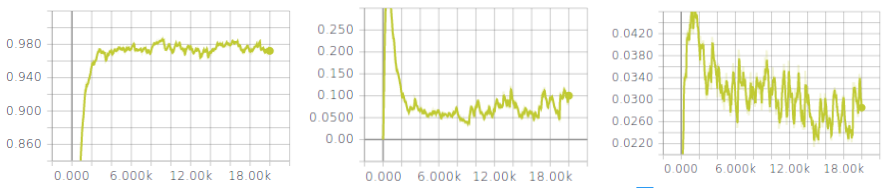


Figure 13: Left: test accuracy, Center: confidence loss, Right: regression loss

5.2 Detection using Flood Fill on a 3D Point Cloud

The approach taken to 3D people detection in this research is to find blobs in the image that are potentially people. This approach is not intended to function well as a single detector, instead serving to solidify detections made by the 2D detector, by making faster but less reliable detections by searching for areas of points the image that are similar in depth. The Pepper robot is able to retrieve 3D data in six different formats called: Yuv, rgb, depth, xyz and distance. In this research the choice was made to use Yuv because is the smallest format available. In this format the data is scaled in range 0 to 255 and returned as a greyscale image, see Figure 14. This approach has the disadvantage of losing the absolute distance from the robot, tests with other formats have shown however that absolute data does not improve the detection rate significantly. Instead the relative distance between points is more important, as it provides most information. One experiment was done were the distance to person candidates was used to filter candidates by determining if the blobs were human sized, however the size proved to inconsistent for filtering to be effective and it would also filter out obscured people.

Figure 14: Yuv greyscale image retrieved from Pepper

5.2.1 Adapting Flood Fill for 3D Data

The flood fill algorithm is a classic algorithm [11] for determining the connected area between some boundary and is usually used in graphical programs to find an area of similar color and fill it. It looks for free areas connected to the initial area and adds them to a list, from those new areas it again finds the next free area until there are no more free areas. To be able to use flood fill on the point cloud data the filling condition for potentially fillable area's needs to be changed from just free to satisfying a certain condition. In the case of 3D data the condition is that the distance between the current area and the next area is smaller than a certain threshold value. Different threshold values were tested and while the scaled nature of the data means that the absolute value of the distance between one point and another varies between images, a distance of 5

generally performs well.

Some adjustments need to be made to the flood fill algorithm in order to efficiently use it for detection. The flood fill algorithm can take significantly longer than usual to complete when the starting point falls within a large field of similar depth such as a wall or a person obscuring the camera. To ensure a predictable running time a threshold is used to stop calculations when a certain amount of fills is reached. A threshold of 2000 points is used as tests show that blobs formed from people never exceed this size. Another adjustment is the addition of a set to keep track of all filled points and return them, these points are needed for determining if a new startpoint has already been included in a previous fill iteration. A full pseudocode description of the 3D flood fill algorithm can be found in Algorithm 1.

Algorithm 1 Flood fill for 3D data

```
1:  procedure FLOOD FILL(image, threshold, (startx, starty), maxiterations,
    fillValue)
2:      indexList = [(startx, starty)]
3:      stack = set(indexlist)
4:      currentIteration = 0
5:      while stack is not free and iteration < maxIterations do
6:          (x, y) = stack.pop()
7:          if (absolute(image[x, y] - originalValue) ≤ threshold then
8:              append (x, y) to indexList
9:          image[x,y] = fillValue
10:         if x then > 0:
11:             stack.add((x - 1, y))
12:         if x then < image.width():
13:             stack.add((x + 1, y))
14:         if y then > 0:
15:             stack.add((x, y - 1))
16:         if y then < image.height():
17:             stack.add((x, y + 1))
18:
        return indexList
```

5.2.2 Flood Fill Candidate Selection

The Flood Fill algorithm on its own does not detect human sized blobs, it only finds the connected area to a given point. To be able to use Flood Fill for detection, flood fill is initialized on different positions in the image and the resulting blobs are filtered on size and shape. The first step is generating the starting points on a grid, Figure 15 left.

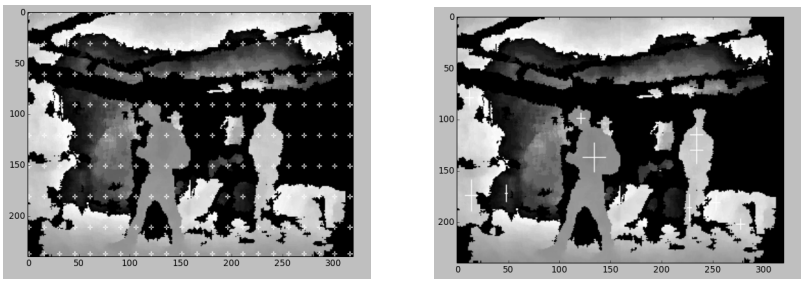


Figure 15: 3D detection. Left: Image with grid, Right: Final Detections

The detection works if at least a single point is initiated on the person to detect. Due to the vertical shape of upright standing humans, the horizontal axis requires a higher density of points. People are generally detectable when they take up at at least a third of the vertical space available, any smaller than that and the shape detected by the camera varies too much and becomes to similar to background to be reliably detected. The next step is to apply Flood fill to all start points. To further speed up the detection the indices retrieved from each fill are stored and used to check if a different starting point is not already filled, start points that have a value of 0, represented as black in all 3D images see Figure 14, are discarded, as 0 is returned when no distance is found. After the fill algorithm is done with expanding a point, a number of conditions are then used to filter the candidate blobs. First blobs consisting of less than 200 indices are thrown away. Next the arithmetic mean and standard deviation of the blob, are calculated. Human blobs are assumed to always have a bigger height than width and the blobs are therefore filtered on having a larger vertical than horizontal standard deviation. The standard deviation is used to remove all indices from the blob that are more than 2 times the horizontal standard deviation from the center, this negates the effect of expansions into for example the floor that could give maximal or minimal values that do not represent the size of the blob well. The points that remain after these conditions have been applied are considered detections and are returned. For a pseudocode version of the algorithm see Algorithm 2.

Algorithm 2 3D People Blob Detection Using Flood Fill

```
1: procedure DETECT BLOBS(maxIterations, gridsize, threshold, fillValue)
2:     startpoints = makeGrid(gridSize.x, gridSize.y)
3:     image = takeImage()
4:     filledPoints = list
5:     detections = []
6:     detectionIndices = []
7:     for startpoint in startpoints do
8:         if image[startpoint.x][startpoint.y] > 0 then
9:             indexList = FloodFill(image, threshold, startpoint, maxIterations, fillValue)
10:            filledPoints.add(indexList)
11:            if length(indexList) > 200 then
12:                standardDeviation = std(indexList)
13:                center = average(indexList)
14:                if standardDeviation.y > standardDeviation.x then
15:                    detections.add(center)
16:        for index in indexList do
17:            if index.x > center.x + std.x * 2 then
18:                detectionIndices.remove(index)
19:            if index.x < center.x - std.x * 2 then
20:                detectionIndices.remove(index)
21:        detectionIndices.add(indexList)
    return detections, detectionIndices
```

5.3 Combining the Detectors

The final step in the combined detection algorithm is the combination of the two different detectors. The convolutional neural network has returned bounding boxes and confidences, while the stereo image detector has returned blobs of indices. The first step in combining these two detections is to transform them to the same coordinate system. As can be seen in Figure 16, the two images get more misaligned the further the pixel is located to the left. This is due to the positioning of the camera on the robot, see subsection 2.2.

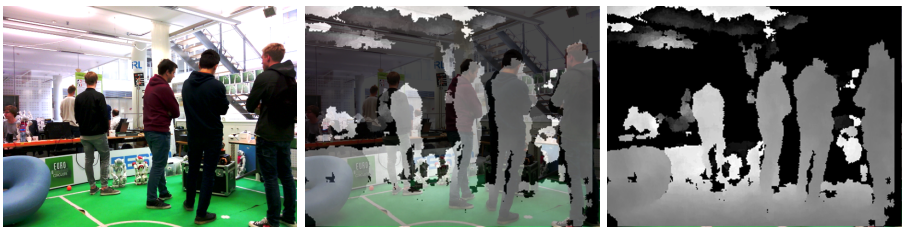


Figure 16: Comparison of the 2D and 3D images

This misalignment is solved by rescaling the 3D points along the horizontal axis. The error of the points increases to the left up to a maximum of around 20 pixels on the 2D image, by redistributing all coordinates according to Equation 3. This solution works well for the level of detail that is provided by the depth sensor,

see Figure 17. Once the detections have been rescaled the information needs to be combined.

$$f(x) = x - \frac{offset}{imagewidth} \cdot x \tag{3}$$

Figure 17: Placing. Left: Uncorrected, Right: Corrected

The first approach is to use the bounding boxes generated by the convolutional neural network as final detections, but filter them using the 3D blobs. Firstly, all centroids that lie within of a CNN bounding box are selected. The percentage of indices of the blob that lie within the bounding box is then calculated, if this is higher than 50% the detection is accepted. This is the strictest method and while it removes up to 99% of all false positives, it also removes all detections where one of the two detectors did not correctly identify the person, significantly impacting recall. Another downside of this method is that the detection range of the 3D blob detector is significantly shorter than that of the CNN, this results in a lot of true positives being filtered because they are to far away to be detected by the 3D blob detector.

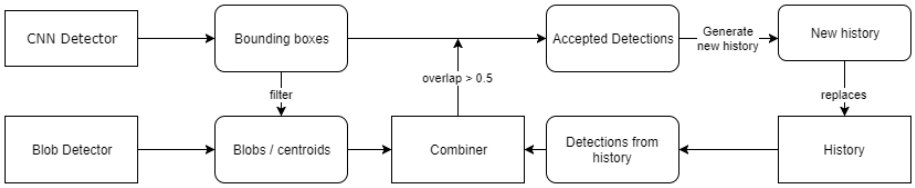


Figure 18: Combined algorithm with history

In the second approach, Figure 18, a detection history is used to stabilize detections. In the current frame all detections are accepted and are placed in a history. All blob locations that are not for at least 50% inside a CNN detection are then checked in the history. If a detection was made in the history at the same location, 50% overlap between the blob and bounding box, as the blob in the current frame, the detection from the history is reused. Additionally it is then also repositioned on the horizontal axis to be centered on the centroid of the blob. This stabilizes the detector by still correctly finding people in the frames where the CNN failed to do so. The vertical axis is not used for repositioning however as the vertical location of the centroid is not as stable as the

horizontal position. This is due to the fact that legs and heads are sometimes not capture by the blob detector, thereby changing the average y position.

6 Evaluation

In this section the different detectors will be evaluated on a new test dataset called the IRL-Students dataset[7]. This dataset was created because there is no existing dataset for this specific configuration of cameras and combination of sensors. This dataset consists of several small sets of 2D and 3D images. The images are taken in the Intelligent Robotics Lab at the University of Amsterdam, where people stood and walked in front of the robot at distances of 0-2, 2-4 and 4-6 meters, for a full description of the dataset see Table 2 and for examples see Figure 19.

Figure 19: Example images from the IRL-lab testset (left to right).
Top: single_2m, single_4m, single_6m.
Bottom: multiple_2m, multiple_4m, crowd_all

Table 2: Data for the IRL-lab test set.

name	distance	n people	n images
single_2m	0-2m	1	105
multiple_2m	0-2m	2	134
single_4m	2-4m	1	101
multiple_4m	2-4m	2	118
single_6m	4-6m	1	150
crowd_4m	0-6m	7	140

[7]Dataset available at `http://uvahome.nl/publications.html`

6.1 CNN Performance

In order to find the baseline performance, the convolutional neural network was tested on the IRL-students dataset. The metrics used for the evaluations are defined as follows:

1. $Precision = \frac{correct\,detections}{total\,detections}$, where correct detections are defined as detections overlapping with the ground truth for at least 50%
2. $IoU\,(Intersection\ over\ Union) = \frac{ground\ truth\ \cap\ detection}{ground\ truth\ \cup\ detection}$
3. $Recall = \frac{detected\ people}{people\ in\ image}$

As can be seen in Table 3, the performance of the CNN is consistent on datasets that contain only 1 or 2 people. It performs slightly worse on datasets that contain people that are close to the camera, the single and multiple 2m sets, in comparison to the sets where people are further away from the camera: single4m, multiple4m and single6m. The precision in these sets stays roughly the same, which means that correct detections are usually centered very well on the person and that detections that overlap just above the 50% threshold occur only rarely. The performance on the crowd set gives a better indication of the weak points of the detector. A sharp drop in recall and precision is seen when many people are crowded together. The problem can likely be attributed to the fact that outlines of people are harder to recognize in these scenes, as people are often obscured by each other. The IoU does not suffer the same loss in performance, indicating that the fewer detections that are made are still centered well on the person.

To conclude, the weak points of the CNN detector are people in close proximity to the camera and detection of people in a large crowd. The first problem can be partly attributed to the scarcity of images that contain people within a 1 meter distance of the camera in the dataset, but also to the fact that the performance of the InceptionV1 network on objects that take up a large section is usually worse as the original network was not designed for the detection of large objects [7]. The second problem can also be contributed to a lack of large overlapping crowds in the dataset, but also to the inherent difficulty of detecting people in crowds[13].

	single2m	single4m	single6m	multi2m	multi4m	multi2-4m	crowd
IoU	0.86	0.89	0.85	0.81	0.88	0.86	0.77
Recall	0.84	1	1	0.63	0.98	0.95	0.54
Precision	0.847	1	0.99	0.83	0.99	0.88	0.68

Table 3: CNN performance on IRL-students dataset

6.2 Blob detector performance

To measure the performance of the 3D blob detector, the metrics used need to be slightly modified. Firstly, IoU cannot be calculated in the same manner as for the CNN, because the detector does not return bounding boxes, but blobs. To still get a sense of how well the blobs are placed on the person, the following

metrics will be defined:

- $Accuracy = \frac{blob\ size}{blob \cup ground\ truth}$
- $Precision = \frac{correct\ detections}{total\ detections}$, where correct detections are defined as blobs of which at least 50% of the indices lie within a bounding box.
- $Recall = \frac{detected\ people}{people\ in\ image}$

The first note here is that precision is still defined using correct people detections. However, it can be said that detections of non humans are not incorrect, because it is not designed to only capture humans, but rather to find all vertical shaped blobs in the data. The results of the detector can be seen in Table 4.

	single2m	single4m	single6m	multi2m	multi4m	multi2-4m	crowd
accuracy	0.98	0.97	na	0.94	0.97	0.98	0.95
recall	0.99	0.87	0	0.97	0.911	0.69	0.64
precision	0.35	0.22	0	0.24	0.25	0.28	0.44

Table 4: Performance of Blob detector on IRL-students dataset

As can be seen, accuracy is consistently high across the datasets where blobs are visible. This indicates that blobs rarely extend outside the ground truth bounding boxes. Recall is high in the instances where people were close enough to the camera, which can be seen by the near perfect performance on the single2m dataset. However the limited detection range reduces the recall, as can be seen by the performance on single4m. Upon visual inspection, Figure 20, it was discovered that people would become undetectable as soon as the distance of 4 meters was even slightly surpassed. Precision stays consistent throughout the dataset, rising slightly in the easiest single2m set, where very little objects were present beside the person. Additionally, precision is higher in the crowd set due to humans obscuring most other objects, making any detected object likelier to be human and thus increasing precision.

6.3 Combined Detection Performance

The goal of the combined detector is to find humans in the given 2D images, with additional information being supplied from 3D images. This means that 2D bounding boxes are the final output of the detector and therefore the same metrics; IoU, precision and recall, will be used to evaluate the output of the detector. The results of the combined detector can be seen in Table 5.

	single2m	single4m	single6m	multiple2m	multiple4m	multiple2-4m	crowd
IoU	0.8	0.85	0.85	0.78	0.881	0.83	0.75
recall	0.85	1	1	0.7	0.983	0.96	0.65
precision	0.85	1	0.99	0.83	0.991	0.86	0.60

Table 5: Performance of the Combined Detector on IRL-students

Generally, a dip in the IoU score can be seen across all datasets, this is likely due to the shape of detections that are obtained from the history. These detections

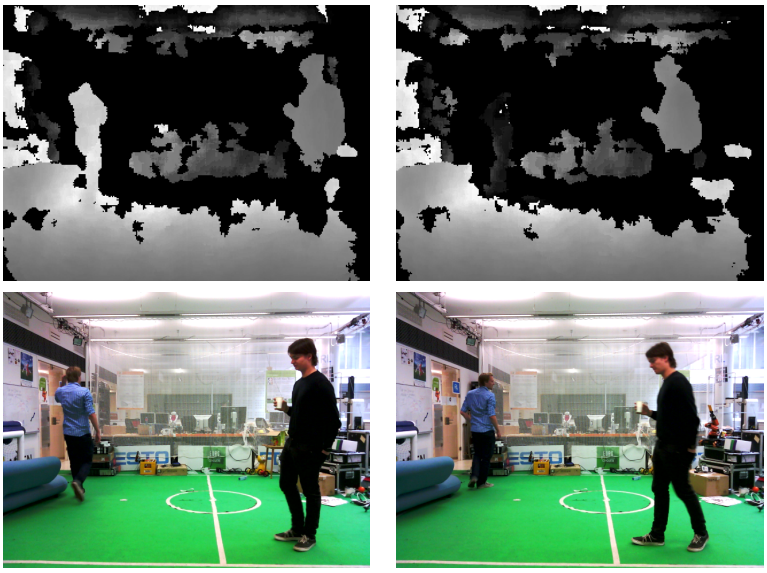


Figure 20: Left Test subject crosses maximum distance measurable by the 3D sensor. Top: 3D images, Bottom: 2D images

are repositioned to the correct horizontal spot, but can still differ in their vertical position. Additionally, a persons bounding box can differ significantly in size if they change their pose. For example an extended arm or a sideways turn can increase or decrease the width of the ground truth bounding box, while the bounding box obtained from the memory would still be sized according to the old pose. The precision stays approximately equal compared to the CNN only based approach, from this can be concluded that the restrictions on accepting histories and the repositioning of the detection accepted from history are strict enough. Finally the recall, while not improving much in the single and multiple datasets, sees a significant increase of 10% in the crowd set. Additionally, performance on the multiple2m dataset was also higher, which on visual inspection seems to originate from a more stable detection on partly obscured people in the dataset. While the recall on the crowd set is still not as high as that obtained from the other sets, it has to be noted that around half of the people in the crowd set are out of range of the 3D sensor. If the 3D blob detector is unable to find human shaped blobs, the history method can not make use of those to detect people. Additionally, a slight delay between the moment the 2D image and the 3D image were taken can be seen in the dataset. This delay causes the person and 3D blob to be in a different position when the person moves across the camera in a fast pace. For an example of a history generated bounding box, see Figure 21.

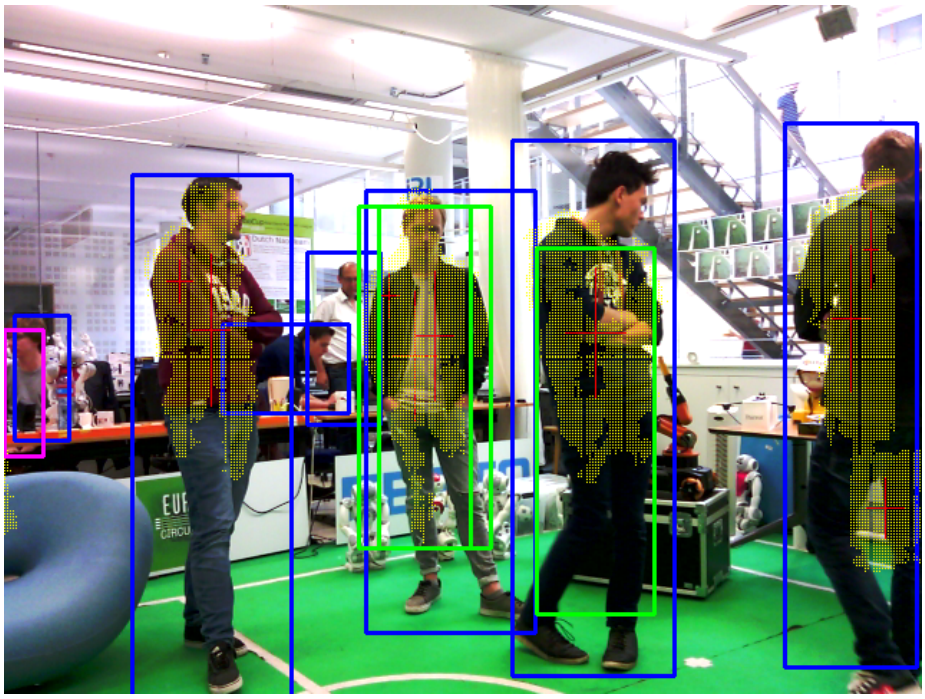


Figure 21: An example where the CNN did not correctly detect people that were detected in the previous frame, but were then detected using the history approach. Ground truth in blue, CNN detections in purple, blob shapes in yellow, blob centroids in red and with detections obtained through the history with 3D blobs in green

7 Discussion

Some concern has been expressed that the use of the InceptionV1 network, while performing well in this research, is not very practical for live detection and classification. Its performance overshadows that of the 3D detector somewhat and it can be said that the performance of the two individual classifiers can not be compared as they do not have the same level of complexity. For example if a convolutional neural network were to be trained on the 3D data interpreted as an image, it would likely outperform the current classifier. However the point that 3D data can provide additional information which can be used to increase performance still stands. Another point of concern is that while the dataset is varied in distance and people density, all images were taken at the same location, making it possible that the results are not representative for other locations.

8 Conclusion

To recall the goal given in the introduction, the aim of this research is to:

> ***Develop*** *a People detection method using the different sensors available to the Softbank Pepper and find the best way of* ***combining*** *the different detection methods.*

To achieve the research goal, the following two questions had to be answered, firstly:

> **Q1** How can the different sensors of the Softbank Pepper be used to detect people?

The combination of the 3D sensor and the 2D cameras was chosen to detect people. A convolutional neural network based approach was taken to detect people in 2D images and a grid based blob detection system based on an expansion algorithm, with filtering was developed to detect people in 3D images. These two approaches were combined using a history based system to reuse earlier detections. This answers the question of how the different sensors can be used to detect people. The second question to be answered was:

> **Q2** Does the inclusion of additional detection methods improve the overall performance in people detection?

In order to measure the performance of the new combined detector, a new dataset called the IRL-students dataset was made. Results show that a CNN detector performs very well on images containing one or two people in the range of 4 to 6 meters from the Pepper, but decreases in performance when people are at a distance of 2 meters to the camera due to the large size of the object to detect. Additionally, it struggles with finding distinct people in large crowds. The blob detector is able to find blobs of people with high recall, but low precision. Combining the two detectors by verifying past detections for the current state using a detection history increases recall for crowded scenes, although the limitations of the hardware do not allow it to detect further than 4 meters. Overall the results indicate that 3D sensor data from the Pepper can be used to improve performance of a convolutional neural network, however a CNN only approach already performs very well for most situations.

An important question now is the performance of the detector in the context of the RoboCup@Home competition. It has been shown that the people detection system described in this paper performs well on different combinations of people density and position. Additionally the test settings used are not unlike the setting of the @Home competition [17], making it likely that performance there would be similar. However the runtime of the detector, while not a concern for this research, is likely to long for use in the competition. However, if the Inception network were to be replaced by a less computationally intensive network, the use of the detector could become more viable.

9 Future Work

A noteworthy addition to this research would be the use of 3D data to separate detections made by the CNN. Some tests done during this research indicate that crowds can be more easily separated into individuals through the use of depth data, as it is easier to recognize a difference in distance in 3D data than to recognize the individuals in 2D images. Another improvement from a practical point of view would be the testing of additional networks that require less computational power, which would make them more suitable for the @Home competition. Finally, additional the use of more complex machine learning techniques, such as clustering algorithms, on the 3D blob detector could further decrease the amount of non-human objects detected and improve detection accuracy.

Acknowledgements

My sincere thanks to Arnoud Visser for his supervision, feedback and his suggestions during the course of researching and writing this thesis. I would also like to thank my fellow lab members, who helped me with difficulties in the implementation, pointed out related research and accompanied me on many coffee breaks. My thanks as well to those who were willing to be in the IRL-students dataset and those that proofread this document.
Finally, I would like to thank my parents for supporting me throughout not only the writing of this thesis, but also throughout my entire study.

References

[1] Thomas Wisspeintner, Tijn van der Zant, Luca Iocchi, and Stefan Schiffer. Robocup@home: Scientific competition and benchmarking for domestic service robots. *Interaction Studies*, 10(3):392 – 426, 2009.

[2] Christian Wojek, Stefan Walk, and Bernt Schiele. Multi-cue onboard pedestrian detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 794–801. IEEE, 2009.

[3] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1014–1021. IEEE, 2009.

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

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[6] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[8] Mauricio Correa, Gabriel Hermosilla, Rodrigo Verschae, and Javier Ruiz-del Solar. Human detection and identification by robots using thermal and visual information in domestic environments. *Journal of Intelligent & Robotic Systems*, 66(1):223–243, 2012.

[9] Vaishak Belle, Thomas Deselaers, and Stefan Schiffer. Randomized trees for real-time one-step face detection and recognition. In *Proceedings of the*

19th International Conference on Pattern Recognition (ICPR'08), pages 1–4. IEEE Computer Society, December 8-11 2008.

[10] Jose Figueroa, Luis Contreras, Abel Pacheco, and Jesus Savage. Development of an object recognition and location system using the microsoft kinect tm sensor. *RoboCup 2011: Robot Soccer World Cup XV*, pages 440–449, 2012.

[11] Theo Pavlidis. Filling algorithms for raster graphics. *Computer graphics and image processing*, 10(2):126–141, 1979.

[12] Christian Wojek and Bernt Schiele. A performance evaluation of single and multi-feature people detection. *Pattern Recognition*, pages 82–91, 2008.

[13] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2325–2333, 2016.

[14] Higinio Gonzalez-Jorge, Belén Riveiro, Esteban Vazquez-Fernandez, J Martínez-Sánchez, and Pedro Arias. Metrological evaluation of microsoft kinect and asus xtion sensors. *Measurement*, 46(6):1800–1806, 2013.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

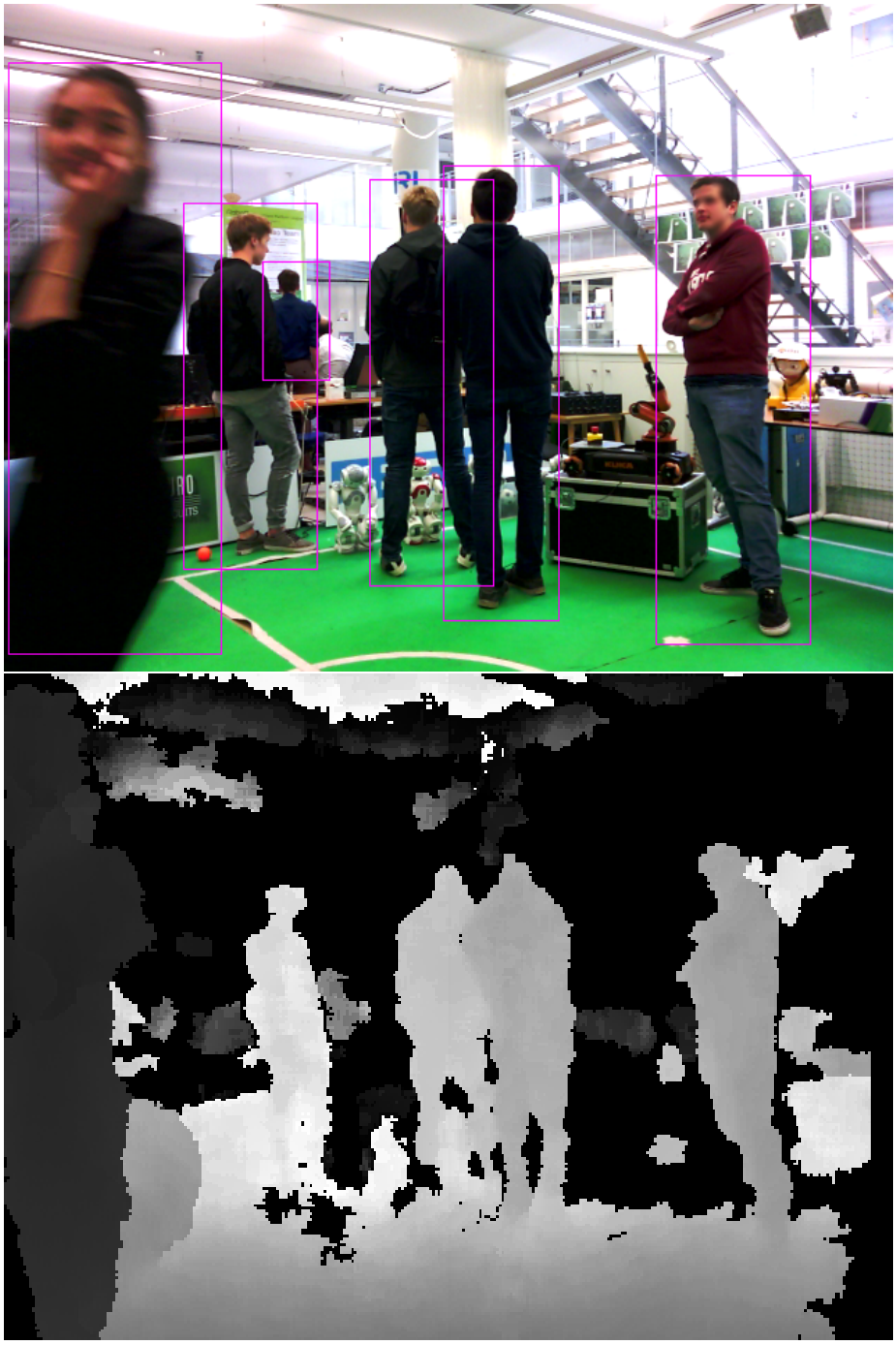
[17] Luca Iocchi, Dirk Holz, Javier Ruiz-del Solar, Komei Sugiura, and Tijn Van Der Zant. Robocup@ home: Analysis and results of evolving competitions for domestic and service robots. *Artificial Intelligence*, 229:258–281, 2015.

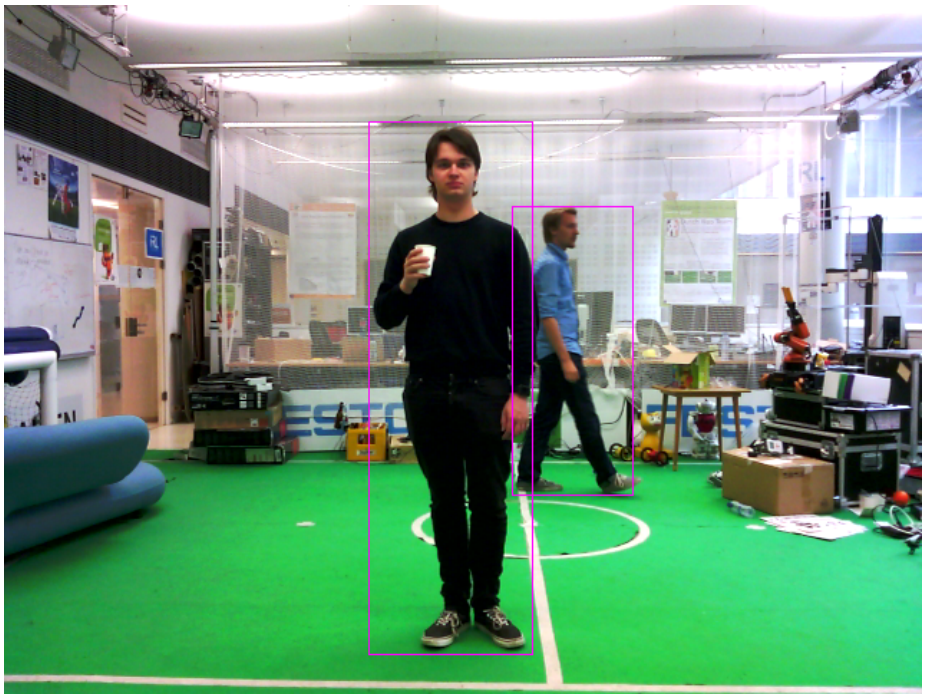
A Appendix

Dataset description

Images for the IRL-students dataset were taken in two different sessions in the Intelligent Robotics lab in the University of Asterdam. Test subjects were asked to participate in the making of a people detection dataset, some people who could not be reached have had their faces blurred for public release. The data recorded as 6 different smaller sets, each one with different content in terms of distance to the test subjects and the number of test subjects. Images were annotated using the sloth annotation tool, Appendix section A. Obscured people were annotated as long as at least half of their body was visible. Additionally when two people cross each other, only a single bounding box was drawn in the frames where one person was entirely obscured. Examples of pairs of annotated color images, bounding boxes in purple, and their 3D counterparts follow on the next pages.

Examples of annotations





Software List

External packages:

- opencv2 `http://opencv.org/`
- numpy `http://www.numpy.org/`
- naoqi `http://doc.aldebaran.com/2-1/dev/community_software.html`
- scipy `https://www.scipy.org/`
- PIL `https://pypi.python.org/pypi/PIL`
- matplotlib `http://matplotlib.org/api/pyplot_summary.html`

Used repositories:

- `https://github.com/cvhciKIT/sloth`
- `https://github.com/TensorBox/TensorBox`
- `https://github.com/tensorflow/models`

Project repositories:

- `https://github.com/jonathan-gerb/UvA-Home`
- `https://github.com/jonathan-gerb/silver-succotash`