Multi-Objective Convolutional Neural Networks for Robot Localisation and 3D Position Estimation in 2D Camera Images

Justinas Mišeikis[1], Inka Brijacak[2], Saeed Yahyanejad[3], Kyrre Glette[4], Ole Jakob Elle[5], Jim Torresen[6]

Abstract— The field of collaborative robotics and human-robot interaction often focuses on the prediction of human behaviour, while assuming the information about the robot setup and configuration being known. This is often the case with fixed setups, which have all the sensors fixed and calibrated in relation to the rest of the system. However, it becomes a limiting factor when the system needs to be reconfigured or moved. We present a deep learning approach, which aims to solve this issue. Our method learns to identify and precisely localise the robot in 2D camera images, so having a fixed setup is no longer a requirement and a camera can be moved. In addition, our approach identifies the robot type and estimates the 3D position of the robot base in the camera image as well as 3D positions of each of the robot joints. Learning is done by using a multi-objective convolutional neural network with four previously mentioned objectives simultaneously using a combined loss function. The multi-objective approach makes the system more flexible and efficient by reusing some of the same features and diversifying for each objective in lower layers. A fully trained system shows promising results in providing an accurate mask of where the robot is located and an estimate of its base and joint positions in 3D. We compare the results to our previous approach of using cascaded convolutional neural networks.

I. INTRODUCTION

With the tendency of robotic hardware becoming cheaper and more powerful, robots are entering our everyday environments. Household robots like vacuum cleaners do not surprise people anymore. Even faster robot adoption happens in hospitals, warehouses and factories. An important reason for this is advancements in environment perception capabilities. Instead of fencing off the robots, the concept of Industry 4.0 is aimed at having a new era of collaborative robots, which are safe to operate in shared workspaces with humans [1]. The concept of a shared workspace has been an active research area for many years, which is still highly relevant today [2] [3]. The industry is catching up to research with robotic platforms like Baxter and Sawyer, which are known to be fully safe to operate around humans. However, they are still at a stage, where collision detection is the main safety system [4]. But we are looking at more sensitive environments, for example, hospitals, where collision detection is not good enough and full avoidance is needed.

[1][4][5][6]Justinas Mišeikis, Kyrre Glette, Ole Jakob Elle and Jim Torresen are with the Department of Informatics, University of Oslo, Oslo, Norway
[2][3] Inka Brijacak and Saeed Yahyanejad are with the Joanneum Research - Robotics, Klagenfurt am Wörthersee, Austria
[5]Ole Jakob Elle has his main affiliation with The Intervention Centre, Oslo University Hospital, Oslo, Norway oelle@ous-hf.no
[1][4][6] {justinm,kyrrehg,jimtoer}@ifi.uio.no
[2] Inka.Brijacak@joanneum.at
[3] Saeed.Yahyanejad@joanneum.at

One of the most common methods to observe the environment is by using vision sensors. In this application, 3D cameras observe the workspace and indicate the areas, which are free of obstructions and are safe to operate in as well as obstacles, which should be avoided. Given this information, a robot can find the safest path to reach its goal. However, normally these sensors are fixed either in relation to the robot or in the environment. In order to function in the same coordinate frame and provide accurate information to the robotic system, Hand-Eye calibration is performed. It works well as long as the setup of the sensors and the robot base stays static. If any of them are moved, intentionally or accidentally, the calibration has to be repeated in order for the sensors to work with the necessary precision. Despite some automatic calibration procedures, the process can still be time-consuming, and the system has to be halted until this issue is resolved [5].

One way to make the environment aware robots is to use long-term environment observation. Such approaches have been used in the development of robot autonomy and self-localisation tasks. This is commonly developed as navigation algorithms for mobile robot platforms to find their way around in the environment and avoid any static or dynamic obstacles on the way. Typically, robot model and dynamics are typically known [6] [7] [8] [9].

Visual-based robot manipulator tracking has been extensively researched as well. End-effector being the main point of focus with the aim of conducting robot control based on visual servoing [10] [11]. Furthermore, it has proven to be an effective method for adaptive redundant robot control in Cartesian space [12]. Image-based tracking of 7-DoF robot arm showed promising results with dynamic parameter tuning as well [13]. In another project, authors use particle swarm optimisation method for fuzzy sliding mode control to track the end-effector of the robot manipulator [14]. Furthermore, robotic arms were combined with deep learning approaches to learn direct motor commands by using visual inputs. They were based on reinforcement learning and by trying thousands of grasps reaching impressive results of adaptive grasping approaches. However, that required many hours of training while using real hardware [15] [16] [17].

One thing that majority of discussed systems have in common is that prior knowledge of the robotic platforms is given or the setups in regards to hardware are fixed. Any changes to the setup would require re-calibration or at least fine-tuning the algorithms to achieve the same level of performance. Furthermore, common obstacle avoidance algorithms for robotic arms are focusing on the end-effector

Fig. 1. Samples from a collected robot dataset. Each row of images represents different robot type in the following order: UR3, UR5 and UR10. The dataset was created using a varying background to provide more robustness.

instead of the whole robot body.

Having non-fixed setup allows easier camera placement in cluttered environments with multiple robots, like a factory floor or automated surgery room. Normally, there is limited space and equipment might have to be shifted around quite frequently. This results in limited line-of-sight or people standing in front of the sensor. Having a multi-camera setup can add the needed redundancy, or using a wearable camera would provide a viewpoint of the operator. On a factory floor, such a camera-based system can give an indication of all the robots located around the person wearing it. A warning or even an emergency stop option can be incorporated into the system for the situations when the robot gets too close to the person within its field of view to ensure a safe operation.

A similar approach could be also used in robot-robot interaction cases, where similar or heterogeneous robots are working in the same environment. Even without having direct communication channels, robots can avoid collisions with each other. On the other hand, this can be used as a redundant navigation system, given the map of the main robots is known, the mobile platform can re-localise itself according to their detected positions. Collaborative tasks would be targeted also, where robots have to hand over tools or work together. Having an active communication channel is not always reliable, so being able to identify robot arms in the environment and their configuration using on-board camera can allow to solve these problems. Provided high enough processing power, swarm robotics could benefit from such systems, where each individual is making independent decisions without any centralised system.

Our current research targets this problem by trying to add flexibility to the robot identification and having easily adjustable setups. One goal is to have a free moving camera and remove the need for Hand-Eye calibration. Instead of having a known transformation matrix between the coordinate frames of the sensor and the robot base, we teach the system to identify the robot body in a 2D color image provided by the vision sensor. This would allow having cameras placed on moving objects, for example, wearable ones or placed on other robots moving in the environment. Our method uses convolutional neural networks (CNNs), which learn visual cues allowing it to understand the environment [18]. The system identifies the robot body in the color image, and depth information normally provided by 3D cameras is not needed for the recognition task anymore. Furthermore, the system estimates the robot body configuration and 3D coordinates of each joint of the robot.

Current work is an extension and improvement of our previously proposed method to use cascaded CNNs (C-CNN) in order to solve this problem [19]. The advantage of using multi-objective CNN is the ability to train the network on multiple tasks simultaneously while re-using the same features instead of having to re-learn some of them when using C-CNNs. Similar multi-objective CNN approaches have been used for detecting facial landmarks, face recognition and localisation as well as orientation [20] [21]. Other similar approaches can be done to optimise the training of the network on two GPUs, each one following each branch [22]. Also, mid-layer parameter transfer between two identical networks, but each one having different sets of objective labels has proven to be effective [23].

This paper is organized as follows. We present the system setup and dataset collection in Section II. Then, we explain the proposed method and CNN architecture in Section III and the training procedure in Section IV. We provide experimental results in Section V, followed by relevant conclusions and future work in Section VI.

II. SYSTEM SETUP AND DATASET COLLECTION

Deep learning typically requires large amounts of diverse training data for robust learning. However, this is an issue for industrial robotics applications, because there are close to none existing public datasets with well-marked ground truth data. Thus, in order to get reliable training data, a new

(a) Color image from the dataset used as an input.

(b) Ground truth model of the robot mask.

(c) Ground truth data of the robot base 3D position in relation to the camera marked on the input image.

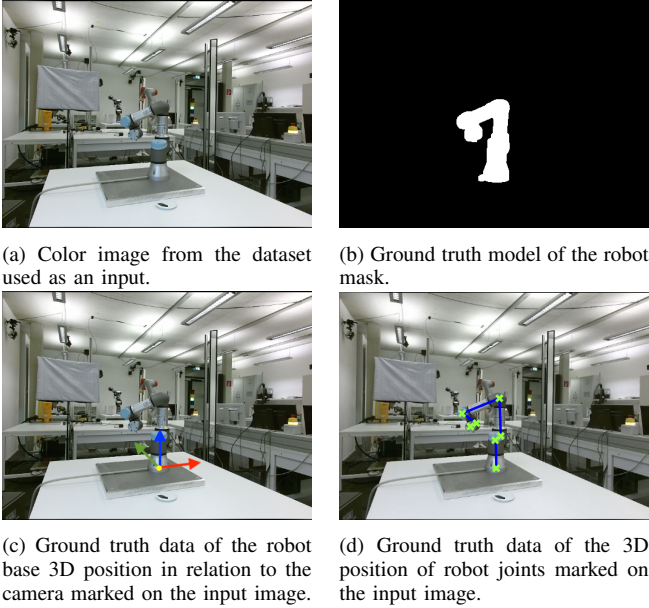(d) Ground truth data of the 3D position of robot joints marked on the input image.

Fig. 2. Example image of the dataset and ground truth examples of the UR3 robot.

dataset was created specifically for the presented application. The whole range of Universal Robots: UR3, UR5 and UR10, were used at three institutions: TU Graz, Joanneum Research and the University of Oslo. All three robots share similar visual appearance, but differ significantly in size, reach and payload capabilities.

As a vision sensor, a Kinect V2 camera is used [24]. It provides both color image and depth information. Depth images are only used for the creation of the ground truth data, while the whole following recognition process is using just a color image as an input.

For each recording, in order to have a precise ground truth data, Kinect was placed at arbitrary position observing the workspace of the robot. At each position, a Hand-Eye calibration was performed by placing a marker on the end-effector of the robot and using both color and depth image for the calibration process [25]. This provides an accurate coordinate frame transformation between the camera and the base of the robot, with an error below 0.52 cm for all the datasets.

TABLE I. Dataset summary describing a number of samples collected for each type of the robot. In total 9 recordings were made, 3 for each type of robot.

Recording	Robot Type	Number of Samples
Rec 1	UR3	211
Rec 2	UR3	252
Rec 3	UR3	463
Rec 4	UR5	252
Rec 5	UR5	756
Rec 6	UR5	1512
Rec 7	UR10	112
Rec 8	UR10	278
Rec 9	UR10	514

Once the transformation is known, a mask defining the location of the robot in the camera image can be calculated. It is done by utilising the encoder information from each joint of the robot and using a simplified model of the

robot. The robot is represented using basic cylindrical and spherical shapes in 3D space according to its model and then mapped onto a virtual 2D image representing the sight of the camera. Thresholding this image results in a robot body mask representation in the camera image. The MoveIt! package was used to implement this method [26].

The robot should be observed from all the different angles and in a high variety of joint angle configurations to achieve good robustness. Movements for the data collection were programmed to provide a high diversity of viewpoints and robot body configurations. Each robot joint is moved through the full range of motion in combination with other joints as well. The step size of the joint movements is varied between the datasets resulting in a different number of samples in each. After each movement, a trigger signal is used to save the data. At each instance, camera images, joint coordinates, Cartesian coordinates of each joint and ground-truth robot mask images were saved. The number of samples per dataset varied from 112 to 1512. The variation was caused by different types and resolutions of programmed robot movements during the data collection. In total 9 datasets were collected, 3 for each type of the robot, summarized in Table I. Example images from the collected dataset are shown in Figure 1. Datasets with UR5 robot were the most extensive given the access to the robot at the lab of the main author. An example of color and ground truth of robot mask, base position and joint positions can be seen in Figure 2.

Recorded images have 512×424 pixel resolution and they are all rectified to compensate for lens distortion. Internal camera calibration was used to ensure that both color and depth information have a good overlap, avoiding any offsets. Random sampling was used to divide the final dataset into the training set and the test set by ratios of 80% and 20% of all the images respectively.

III. METHOD

Our approach is based on a multi-objective CNN structure. This approach allows us to get multiple outputs of different types by having just one input. It is achieved by having a number of convolutional layers, which are common for the whole system and then branching out the structure for each of the objectives. The whole system is trained simultaneously, meaning that the features in common layers are reused.

In our case, we train for four objectives:

- Robot mask in the image
- Robot type
- 3D Robot base position in relation to the camera
- 3D Position of the robot joints

The structure of the CNN is shown in Figure 3. It consists of the two main branches. The first one learns a classification task of finding the robot in the input image. It results in a robot mask defining the location of the robot. The second branch is for the regression tasks of finding the 3D robot base coordinates in relation to the camera and the 3D coordinates of each of the robot joints. In addition, on the same branch, the classification of the robot type is done.
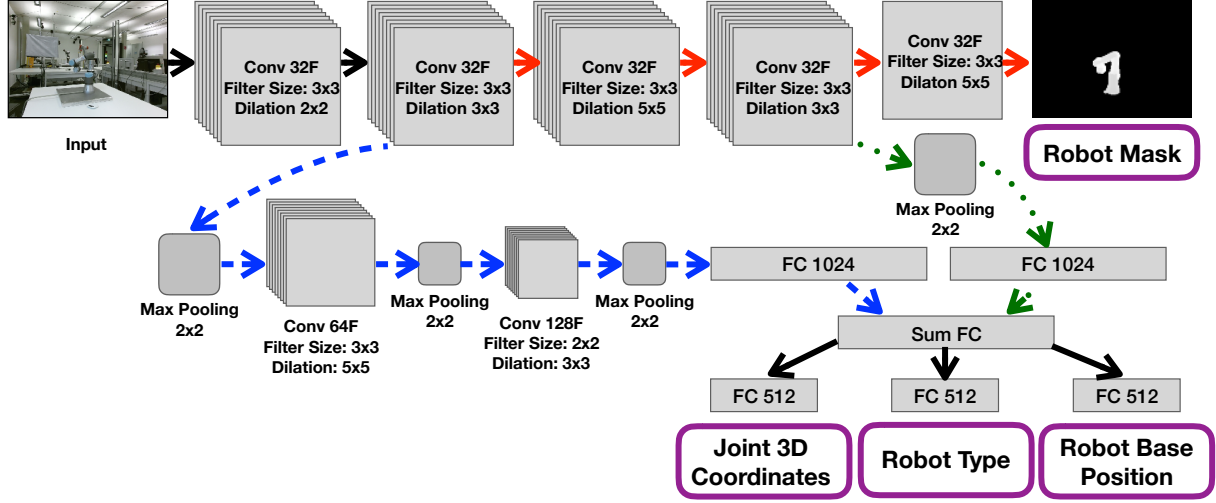
Fig. 3. Multi-objective CNN structure. Input is a simple 2D color image and the network is trained for four outputs: robot mask, 3D coordinates of robot joints, 3D coordinates of robot base position and robot type. There are two main branches of the CNN. The first one is aimed to learn the features leading to an accurate robot mask mainly consisting of dilated convolutional layers. It is marked by red solid arrows. The second branch, marked in blue dashed arrows, consists of a number of max pooling and dilated convolutional layers with fully connected layers at the end. The goal is to predict coordinates as a regression task as well as classify the robot type. Additionally, there is a branch starting from the 4th convolutional layer of robot mask task to the end of the blue branch using summing of fully connected layers, marked in dotted green arrows. It adds the information of features well defining the visual representation of the robot to the other tasks further improving the results. The whole CNN is trained for all four outputs simultaneously using a common loss function.

In addition, there is the second branch from the 4th convolutional layer towards the robot mask, which connects to the second branch. Given the idea that robot body parts are learned quite well for the robot mask classification task, this additional input provides the essential information for identifying the location of the robot joints. Fully connected layers, which are summed, are believed to filter the important visual cues and assist for the coordinate regression tasks.

A. Loss Functions

Loss functions are used to determine the quality of training. Given we have four objectives, we first describe loss functions for each one. Because the network is trained for all of the objectives simultaneously, finally we combine all four loss function into one used for the actual training.

The loss function for the robot mask was designed to adjust for a small area the foreground object takes up in the input image. In our datasets, the area taken up by the robot body in the input image was varying between $6-17\%$ of the whole image. If the loss function does not compensate for this, the CNN could classify all the pixels as background and still achieve the accuracy of 83 to 94%, which is conceptually wrong. To prevent this, the foreground weight w_{fg} is calculated, as described in Equation 1. It is based on the inverse probability of the foreground and background classes, where $Y \in \{fg, bg\}$.

$$w_{fg} = \frac{1}{\mathsf{P}(Y = fg)} \quad (1)$$

The background weight w_{bg} is calculated in Equation 2.

$$w_{bg} = \frac{1}{\mathsf{P}(Y = bg)} \quad (2)$$

The robot mask loss function is calculated in two steps. First, a per-pixel loss l^n is calculated in Equation 3, where i_{est} is $\mathsf{P}(Y = fg)$, $(1 - i_{est})$ is $\mathsf{P}(Y = bg)$ and i_{gt} is the ground truth value from the mask image.

$$
\begin{aligned}
l^n(I^n_{est}, I^n_{gt}) = &- w_{fg} i_{est} \log(i_{gt}) \\
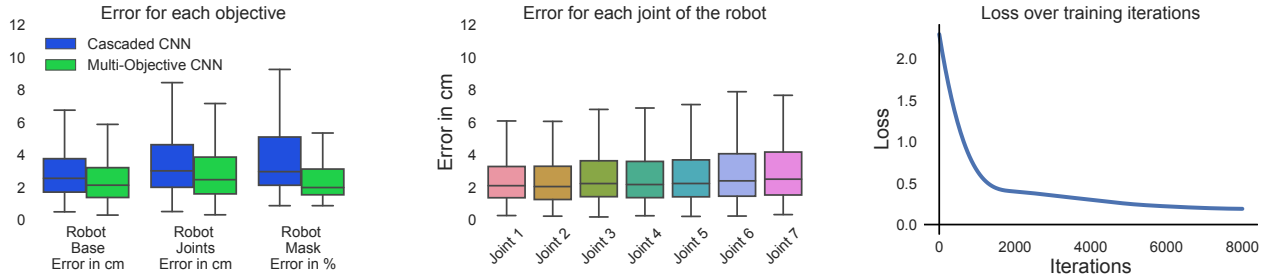&- w_{bg}(1 - i_{est}) \log(1 - i_{gt})
\end{aligned} \quad (3)
$$

Then, it is used as an input to calculate normalised loss for the whole image $\mathcal{L}_{mask}$ in Equation 4. A normalisation factor $\mathcal{N}$, which is a number of pixels in the image, allows us to keep the learning rate fixed, despite the variance of the input image size.

$$\mathcal{L}_{mask}(I_{est}, I_{gt}) = \frac{1}{\mathcal{N}} \sum_n l^n(i_{est}, i_{gt}) \quad (4)$$

Loss functions for both robot base coordinates and the coordinates of the robot joints are formulated as regression tasks. Both of them use Euclidean distance between estimated and ground truth values. Loss function for the 3D coordinates of robot joints $\mathcal{L}_{Jcoords}$ is described in Equation 5, where N_j is the number of joints, J_i defines ground truth position of each joint and E_i is the estimated position of each joint by the CNN.

$$\mathcal{L}_{Jcoords} = \frac{1}{N_j} \sum_{i=1}^{N_j} \|J_i - E_i\|_2 \quad (5)$$

Similarly, the loss function for the coordinates of the robot base $\mathcal{L}_{Bcoords}$ is shown in Equation 6. B_{xyz} is the ground truth position of the robot base in 3D and E_{xyz} is the estimated 3D position of the robot base. These positions are relative to the camera. Considering the goal of detecting the position of the robotic manipulator, estimating just Cartesian

(a) Comparison of the Multi-Objective CNN approach against the Cascaded CNN approach [19]. Errors for all the objectives are smaller, as well as the range of quartile values.

(b) Error for the 3D coordinate estimation for positions of each robot joint. It can be seen that the error slightly increases for joints further away from the base.

(c) Value of the loss function on the test set over iterations during the training process.

Fig. 4. Evaluation of our method by testing the trained system on the test dataset.

coordinates is sufficient. If necessary, the angles of each joint in relation to the robot base could be calculated by using coordinate positions.

$$\mathcal{L}_{Bcoords} = \left\| B_{xyz} - E_{xyz} \right\|_2 \qquad (6)$$

The loss function to identify the robot type was defined as a categorical cross-entropy problem. It is commonly used for multi-class classification problems. $\mathcal{L}_{type}$ is calculated in Equation 7, where p is the ground truth labels, q are the predicted labels and $c \in R$, where R are all the available types of robots in the dataset.

$$\mathcal{L}_{type} = -\sum_c p(c) \log q(c) \qquad (7)$$

The final loss function $\mathcal{L}_{final}$ is a weighted combination of all four previously described functions. The larger the weight W, the higher the emphasis on the correct prediction of the corresponding value. And the weights should be selected to have a good overall performance of the system. The calculation of $\mathcal{L}_{final}$ is described in Equation 8.

$$\mathcal{L}_{final} = W_{mask}\mathcal{L}_{mask} + W_{Jcoords}\mathcal{L}_{Jcoords} \\ + W_{Bcoords}\mathcal{L}_{Bcoords} + W_{type}\mathcal{L}_{type} \qquad (8)$$

In order to keep the CNN easily adaptable to other types of robots in the future, no prior information about the robot model is incorporated in the system. The raw CNN output is used to evaluate the accuracy of the results.

IV. CNN TRAINING

Training of the multi-objective CNN is done for all four objectives at the same time. One possibility to adjust the quality of results is to adjust the weights given to the loss functions of each of the objectives when defining the final loss function of the system. In our case, the weight values were hand-selected using trial and error during the testing phase. Selected weight values were the following:

- W_{mask}: 1.0
- $W_{Jcoords}$: 1.5
- $W_{Bcoords}$: 1.5
- W_{type}: 0.3

The training is done on the training set, including images of all three types of robots simultaneously. In total 926 samples for UR3, 2520 samples for UR5 and 904 samples for UR10 were used

In order to speed up the process and have reasonably sized mini-batches, the input size of the images was reduced by half from the original dimensions, down to 256×212 pixels. The pixel intensity values of the input images were normalised to the range between 0 and 1. Furthermore, pixel values of the ground truth images are clipped to avoid division by zero in cases when the estimated mask fits the ground truth perfectly. In order to avoid any training biases, the data were randomly shuffled and split into mini-batches of 64 images each, fully utilising the memory of the GPU. The learning rate was set to 0.001 at the beginning of the training and then gradually decreased to 0.000001 as the training progressed. The CNN converged after 8000 iterations. It took 60 hours to train the system using a regular NVIDIA GeForce 1080 GTX graphics card.

V. RESULTS

The evaluation was done by testing our network on the test set and comparing the output against the ground truth data. The robot mask accuracy is defined by comparing a number of pixels in the CNN output image that match the ground truth mask. For the robot joint and base coordinates, Euclidean distance between the CNN estimated results and ground truth results was calculated. Robot type accuracy was computed by counting the percentage of correct classification instances. We compare the results against our previously presented C-CNN approach [19].

Robot mask classification achieved an accuracy of 98%, which is almost 3% improvement compared to our previous method, as seen in Figure 4(a). A significant amount of this error comes from failing to estimate sharp corners in the mask image because CNN outputs slightly blurry mask compared to ground truth. It is likely that some post-processing would allow even further improvement by increasing the sharpness of the mask.

The overall error of the 3D position of robot joints was 3.16 cm, which is a slight improvement compared to the error of 3.32 cm in our previous work. If we analyse each
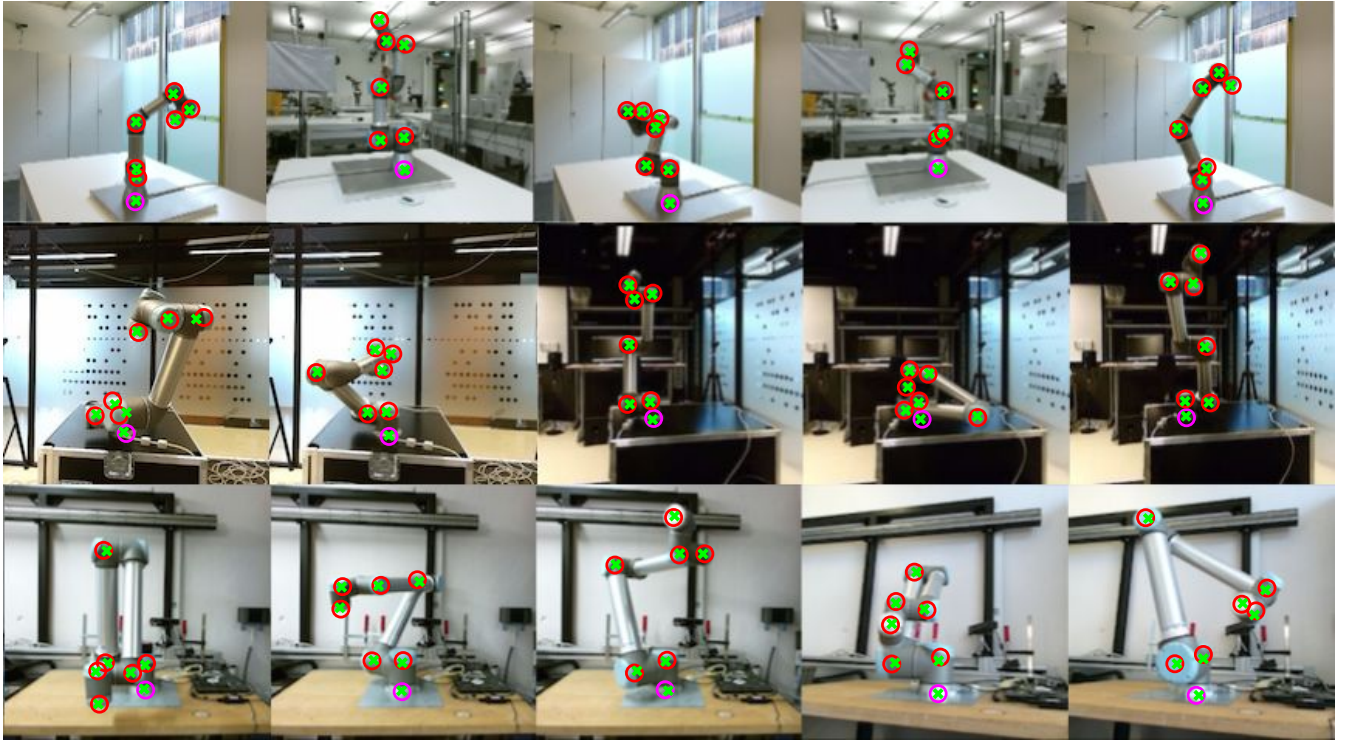
Fig. 5. Estimated robot joint position coordinates marked on the images taken from the dataset. Due to difficulty in visualising 3D coordinates on printed figures, the estimated joint coordinates were mapped back into 2D images. Green crosses indicate the ground truth position, red circles indicate predicted positions of joints and magenta circles indicate the predicted position for the robot base.

joint separately, we can see the tendency of the joints closest to the robot base having a smaller average error, as well as smaller scatter, compared to the joints closer to the end-effector. Results are showing that in Figure 4(b). It can be explained by analysing the reachability from the base of each of the joint. The end effector has the largest range of motion, and it reduces for the joints closer to the robot base. This means the range of possible positions varies significantly, and estimation is more difficult in the larger range of possible positions. However, the error difference is minor.

TABLE II. Summary of the results on the test set of a Multi-Objective CNN with a comparison to our previous work using C-CNN.

Measure	Current Work	Previous Work
Mask Accuracy, %	**98%**	94.6%
Robot Type Accuracy, %	**98.3%**	—
Joint Pos Error (Mean)	**3.16cm**	3.32cm
Base Pos Error (Mean)	**2.74cm**	2.97cm

The estimation of the position of the robot base in relation to the camera had an average error of 2.74 cm. Once again, this is lower compared to C-CNN approach, where the same estimation error was 2.97 cm. Robot type classification made just a few wrong decisions resulting in 98.3% accuracy. The forward propagation time (detection speed) of the neural network was on average 15 ms for one image, making it suitable for real-time applications.

The final results are summarised in Table II, and the estimated coordinates by the full system marked over the dataset images can be seen in Figure 5. Because it is difficult to show 3D estimations on 2D figures, the visualisation of estimation is done by mapping the estimated 3D coordinates back onto input images.

Both in the current multi-objective CNN approach and the C-CNN method, we used exactly the same datasets for training and testing, so the results can be compared directly. Given the lack of similar work, no suitable benchmark was found to allowing a direct comparison of achieved results.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a solution for detecting a robot manipulator and estimating the positions of its joints in a 2D camera image. A camera can be placed in arbitrary positions overlooking the robot workspace and the method successfully localizes the robot without the need for any additional setup or Hand-Eye calibration. This provides more flexible and quickly reconfigurable environment aware robotic setups for tasks like human-robot or robot-robot interaction. We have used three types of robots produced by Universal Robot for training and testing of the system: UR3, UR5 and UR10.

Our system uses a multi-objective convolutional neural network approach to achieve the goal. It optimises the system for four objectives simultaneously provides the mask of an area where the robot is present in the camera image, its base position in relation to the camera, 3D positions of the joints of the robot as well as the type of the robot, respectively the 3D joint position error was less than 3.16 cm, the robot mask accuracy was 98% and the robot type was successfully recognised in over 98.3% of cases. These results are an improvement of our previously presented C-CNN approach, both in accuracy and flexibility of the system.

Given current results, the continuation of work will be to apply this method in more complex environments containing multiple robots and people working in the same workspace. Self-occlusions were present in the tested datasets and some minor occlusions of other objects, however, more evaluation is needed using cases like people or other machinery passing by between the camera and the robot blocking the view.

This work has multiple possible applications. One would be the safety aspect of identifying robots in robotised environments like factory floors, warehouses or automated surgery rooms where an operator has a wearable camera detecting robots in the field of view. Another application would be for robot-robot interaction. With swarm robotics, both homogeneous and heterogeneous, and different sizes, direct communication between them is not always reliable. Our approach would allow the robots to observe and track each other using small cameras and identify the intentions of other robots in the surroundings.

For the human-robot collaboration tasks, a person tracking can be achieved using devices like Leap Motion or skeleton tracking to estimate of the relative hand positions to the robot. This can be used for tool handover between the person and the robot, working towards a common goal or even hand-gesture control, while avoiding any unwanted physical contact between the two.

In the future, we plan to test the system with more types of the robots by using transfer learning on pre-trained CNN. In this case, the dataset needed to teach to identify a new robot type should be significantly reduced compared to the current setup. Adding human skeleton tracking would move the work closer to the real-world human-robot interaction tasks. The system will be tested in some use case scenarios to identify the robustness in less controlled environments with more illumination changes and changing setups.

ACKNOWLEDGMENT

REFERENCES

[1] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015.

[2] J. Roach and M. Boaz, "Coordinating the motions of robot arms in a common workspace," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 437–444, 1987.

[3] J. Leitner, S. Harding, M. Frank, A. Förster, and J. Schmidhuber, "Transferring spatial perception between robots operating in a shared workspace," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1507–1512.

[4] C. Fitzgerald, "Developing Baxter," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–6.

[5] J. Miseikis, K. Glette, O. J. Elle, and J. Torresen, "Automatic calibration of a robot manipulator and multi 3d camera system," in *System Integration (SII), 2016 IEEE/SICE International Symposium on*. IEEE, 2016, pp. 735–741.

[6] S. Schneegans, P. Vorst, and A. Zell, "Using RFID Snapshots for Mobile Robot Self-Localization." in *EMCR*, 2007.

[7] J.-S. Gutmann and C. Schlegel, "Amos: Comparison of scan matching approaches for self-localization in indoor environments," in *Advanced Mobile Robot, 1996., Proceedings of the First Euromicro Workshop on*. IEEE, 1996, pp. 61–67.

[8] O. Stasse, A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. Kheddar, "Real-time (self)-collision avoidance task on a hrp-2 humanoid robot," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 3200–3205.

[9] A. De Santis, A. Albu-Schaffer, C. Ott, B. Siciliano, and G. Hirzinger, "The skeleton algorithm for self-collision avoidance of a humanoid manipulator," in *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*. IEEE, 2007, pp. 1–6.

[10] W. J. Wilson, C. W. Hulls, and G. S. Bell, "Relative end-effector control using cartesian position based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 684–696, 1996.

[11] A. Ruf, M. Tonko, R. Horaud, and H.-H. Nagel, "Visual tracking of an end-effector by adaptive kinematic prediction," in *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 2. IEEE, 1997, pp. 893–899.

[12] B. Daachi and A. Benallegue, "A neural network adaptive controller for end-effector tracking of redundant robot manipulators," *Journal of Intelligent & Robotic Systems*, vol. 46, no. 3, pp. 245–262, 2006.

[13] I. Siradjuddin, L. Behera, T. M. McGinnity, and S. Coleman, "Image-based visual servoing of a 7-DOF robot manipulator using an adaptive distributed fuzzy PD controller," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 2, pp. 512–523, 2014.

[14] M. R. Soltanpour and M. H. Khooban, "A particle swarm optimization approach for fuzzy sliding mode control for tracking the robot manipulator," *Nonlinear Dynamics*, vol. 74, no. 1-2, pp. 467–478, 2013.

[15] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3406–3413.

[16] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3389–3396.

[17] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[18] P. Y. Simard, D. Steinkraus, J. C. Platt, *et al.*, "Best practices for convolutional neural networks applied to visual document analysis." in *ICDAR*, vol. 3, 2003, pp. 958–962.

[19] J. Miseikis, P. Knobelreiter, I. Brijacak, S. Yahyanejad, K. Glette, O. J. Elle, and J. Torresen, "Robot Localisation and 3D Position Estimation Using a Free-Moving Camera and Cascaded Convolutional Neural Networks," *ArXiv e-prints*, Jan. 2018.

[20] X. Yin and X. Liu, "Multi-task convolutional neural network for pose-invariant face recognition," *IEEE Transactions on Image Processing*, 2017.

[21] Y. Wu, T. Hassner, K. Kim, G. Medioni, and P. Natarajan, "Facial landmark detection with tweaked convolutional neural networks," *arXiv preprint arXiv:1511.04031*, 2015.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[23] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[24] P. Fankhauser, M. Bloesch, D. Rodriguez, , R. Kaestner, M. Hutter, and R. Siegwart, "Kinect v2 for mobile robot navigation: Evaluation and modeling," in *IEEE International Conference on Advanced Robotics (ICAR) (submitted)*, 2015.

[25] T. Heikkilä, M. Sallinen, T. Matsushita, and F. Tomita, "Flexible hand-eye calibration for multi-camera systems," in *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2000, pp. 2292–2297.

[26] I. A. Sucan and S. Chitta, "MoveIt!" *Online Available: http://moveit.ros.org*, 2013.