Pose Estimation of a Robot Arm from a Single Camera

Kiruthikan Sithamparanathan
Department of Electrical Engineering
University of Moratuwa
Moratuwa, 10400, Sri Lanka
kiruthikan.s@outlook.com

Sarangan Rajendran
Department of Electrical Engineering
University of Moratuwa
Moratuwa, 10400, Sri Lanka
saranganr@outlook.com

Pirakash Thavapirakasam
Department of Electrical Engineering
University of Moratuwa
Moratuwa, 10400, Sri Lanka
pirakashthavapirakasam@outlook.com

A.M. Harsha S. Abeykoon
Department of Electrical Engineering
University of Moratuwa
Moratuwa, 10400, Sri Lanka
harsha@uom.lk

***Abstract*— This paper describes a vision based deep learning approach to estimate the pose of a robot arm from a single camera input, without any depth information. Conventionally, pose of robot arm is determined using encoders which sense the joint angles, and then the pose of each link (including the end effector) relative to the robot base is obtained from the direct kinematics of the manipulator. But there may be inaccuracies in the determined pose when the encoders or the manipulators are malfunctioning. This paper presents an approach based on computer vision, where a single RGB camera is fixed at a distance from the robot arm. Based on the kinematics of the manipulator and the calibrated camera, the input 2-dimensional image is reconstructed in 3-dimensional form and the pose of the manipulator is determined by means of a deep network model trained on synthetic data. Furthermore, a graphical user interface (GUI) is developed, which simplifies the output interpretation for users who operate the implemented system. Finally, the effectiveness of the proposed approach is demonstrated via several examples and results are presented. The proposed approach cannot entirely replace the function of encoders. Instead, it can be treated as a backup method which provides a reference solution.**

Keywords—— robot arm, pose estimation, 3D object reconstruction, convolutional neural network, deep learning

I. INTRODUCTION

A robot arm is a system of serially connected links that are moved by rotating about the joints actuated by motors. It is considered to be moving in a three-dimensional workspace, and the term 'pose' is defined as the combined position and orientation of all of its links at a given instance. Each joint has an encoder mounted inside to measure the degree of actuation. For a rotary joint, this encoder-measured value is called 'joint angle'. Therefore, for a robot arm consisting only of rotary joints, pose essentially becomes the combination of all the joint angles at a given time. It is fundamental for fulfilling the feedback requirement of motion control. When the kinematics of the robot arm and the current joint angles are known, the pose of the robot arm, and thus the position and orientation of the end effector relative to the base of the robotic manipulator can be computed. However, getting good estimates for these angles can be difficult due to inaccuracies such as joint frictions or malfunctioning encoders. Therefore in many robotic systems, there exists a gap between where the robot estimates its arm to be and where it really is. This becomes a major issue especially in manipulation tasks. Also, in situations where incremental encoders are used instead of absolute encoders for robot arms, there will be a need to know the previous pose in order to determine the current pose. Therefore, additional techniques for estimating the robot arm's pose are considered necessary. This paper proposes a visual approach to figuring out the pose of a robotic manipulator from an RGB image taken by a single camera.

An RGB image which is 2-dimensional cannot inherently describe the pose, which is a 3-dimensional quality. Most of the researches addressing pose determination of a robot arm by visual methods rely on additional means to bridge this lack of depth information. Most popular approach is to attach markers on the surface of the robot arm or create track points on images using edge detection algorithms and track them to estimate the pose [1][2][3]. The main limitation of this kind of approach is that these markers or track points must always be in sight of the camera to predict the pose effectively, which constraints the estimable poses of the robot arm in space.

Bohg *et al.* overcame the need for markers by using RGB-Depth cameras detect the robot arm in 3D space [4]. In their method, the estimation problem was transformed into a classification problem where each pixel in a depth image was classified to be either robot or background and was input to train a random decision forest. Later on, they employed a random regression forest trained on synthetically generated data for the pose estimation task [5]. This method also used depth images to estimate the angular joint positions without any prior knowledge from the joint encoders, and the approach also worked on real depth images. While this solution is highly competent, it still requires depth information that cannot be supplied by a regular RGB camera that outputs 2D images.

Deep Learning has recently been growing as the preferred mode of research for studies that involve large volume of data, particularly images, as it is capable of solving 2D-image tasks such as image classification, object detection, semantic segmentation, etc. Few researchers have utilized it for the problem of robot arm pose estimation too. Miseikis *et al.* had collected some datasets and trained the multi-objective network for robot pose estimation and localization tasks [6]. Recent studies have shown that deep learning can also be employed to predict 3D volume of objects from 2D images with moderate success. Lin *et al.* proposed a framework to efficiently predict the 3D structure of an object in the form of point clouds using 2D convolutions [7]. Since a 3D volumetric prediction adds a new dimensional information which was not present 2D image input, it opens an avenue of research potential to employ it for the pose estimation problem, which has not been tested till now.

This paper proposes a simpler, modified approach to predict 3D volume from a 2D image of the manipulator and use it to estimate the pose. Lin *et al.* had tested and demonstrated their model on everyday domestic objects whose models are imported from ShapeNet repository [7]. Since the target object in our case is more specific, we opted to model our robot arm virtually and create our own synthetic database of poses. In our approach, the robot arm extracted from a 2D RGB image is reconstructed as a 3D point cloud using a Perspective Generator pipeline. Two different datasets, one as 2D pose images and other as vertices of the 3D surface of robot arm, are generated and trained upon to achieve this volumetric prediction. This reconstructed point cloud is then input to a pose estimator network to predict the joint angles. This proposed method cannot entirely replace encoders but can be treated as a reference solution to validate the accuracy of joint angle readings when there are possible malfunctions in the encoders or the manipulator.

This paper is organized as follows. Section II describes the modelling of the robot arm manipulator and its kinematics required for the synthetic generation of the pose image database. Section III presents the proposed framework to determine the pose of the robot arm manipulator. Section IV describes the types of input data used and how they are prepared. Section V evaluates the model with experiments and shows their results. Section VI concludes the contributions of this study.

II. Modelling of the Manipulator

In order to demonstrate the proposed approach, a 'KUKA KR 6 R900 sixx' robot arm manipulator is considered in this study. It has six degrees of freedom, and all joints are revolute. The six joints and their axis of rotations are as illustrated in Fig. 1, where the positive and negative directions of each of the joint angle is marked.

The relationship of one joint axis frame's position and orientation with respect to a previous joint axis frame is described by transformation matrices. By post-multiplication of successive transformation matrices, it is possible to relate the position and orientation of any point in the robot arm with respect to the base frame of the robot arm manipulator.

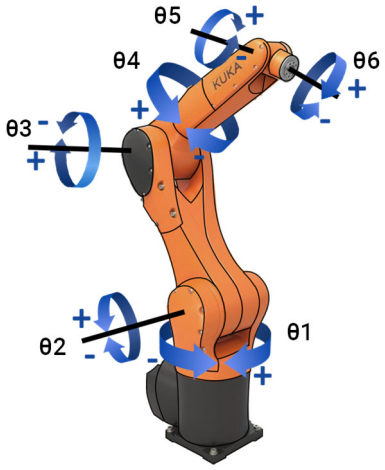


Fig. 1. Joint Angles in selected KUKA Robot Arm

In order to train the deep learning model to identify a robot arm and figure out its pose from a camera image, a large database of images from the camera viewpoint with different poses of the robot arm should be created. Considering the volume of the input database, and repeatability of the process, it was decided to model the robot arm manipulator synthetically in a virtual environment. Recent studies have shown that synthetically generated images can be successfully used as training input data for a real-world application with minimal deviations in expected output [5][8]. In the study conducted by Lee *et al*, it was shown that by training a neural network with images of synthetically modelled robot arm, it was possible to find an external camera's transformation matrix with respect to a real robot arm [8]. In our study, each link of the selected robot arm manipulator was modelled in Blender® software as separate independent objects and were applied motion constraints derived from transformation matrices to mimic real joints.

III. Proposed Approach

Overview of the system is illustrated in Fig. 2. Table I contains the associated nomenclature. The system consists of three main components: perspective generator, pose estimator and optimizer.

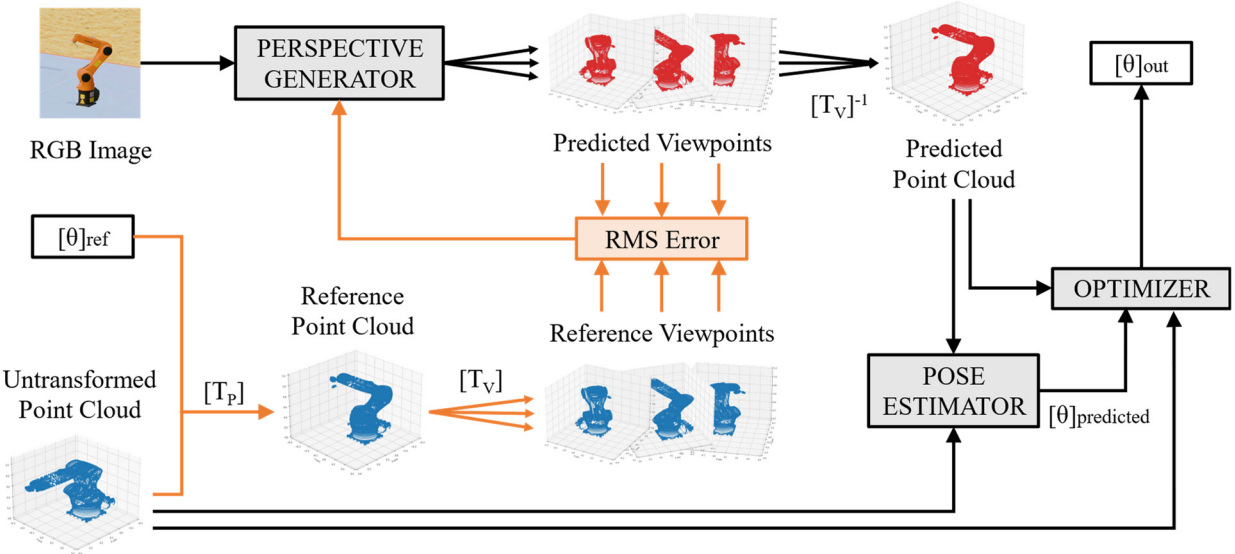


Fig. 2. System Overview. Processes marked with orange arrows are part of the training stage of the model and are not used once the model is trained.

TABLE I
NOMENCLATURE

Symbol	Description
$[\theta]_{ref}$	Reference Joint Angles (input during training)
$[\theta]_{predict}$	Joint Angles Predicted by Pose Estimator
$[\theta]_{out}$	Optimized Joint Angles (final output)
$[T_P]$	Pose Transformation Matrix
$[T_V]$	Viewpoint Transformation Matrix
$[T_V]^{-1}$	Inverse of Viewpoint Transformation Matrix
$[P]_{initial}$	Untransformed Point Cloud at home position
$[P]_{ref}$	Transformed Point Cloud at desired pose
$[P]_{predict}$	Predicted Point Cloud at desired pose

Perspective generator component takes the 2D RGB image as input and predicts a 3D representation of the robot arm at multiple viewpoints. The suitable form of 3D representation was found to be point cloud, where the vertices of the robot arm's outer surface make up the points, as shown in Fig. 3. Compared to other forms of 3D representation such as voxels, point cloud is light on memory and computation [7]. The predicted point cloud is then sent through pose estimator component where the joint angles are estimated through linear regression. The optimizer component improves the accuracy of the model in predicting the joint angles.

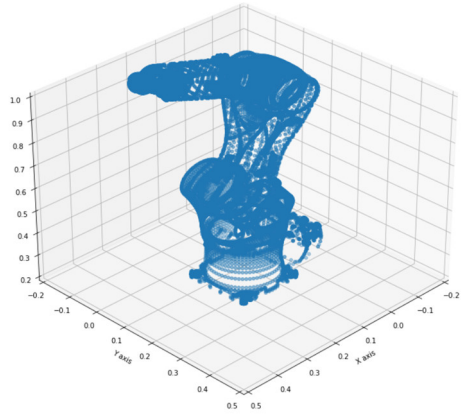


Fig. 3. 3D representation of the robot arm

As indicated in Fig. 2, a point cloud of the robot arm in its home position must be input to the system initially, to be considered as the untransformed point cloud ($[P]_{initial}$). Home Position refers to the robot configuration where all joint angle values are zero.

A. Perspective Generator

This is a fully convolutional neural network (CNN) as illustrated in Fig. 4, whose input is an image which is stored as an array of size h×v×3, where h and v denote the horizontal and vertical pixel resolutions of the image, and 3 indicates the RGB colour channel information of each pixel. The output of the CNN is a set of 3D point clouds of the same robot arm configuration observed from different viewpoints. This is stored as an array of size n×m×3, where n denotes the number of viewpoints, m denotes the number of points in the robot arm point cloud, and 3 indicates the cartesian coordinate information of each point.

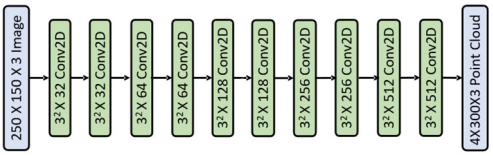


Fig. 4. Convolutional Neural Network Architecture

In order to train the perspective generator to predict the viewpoints correctly from an image, the reference viewpoints corresponding to the robot arm with the same pose as in the image must be given. The generation of these reference viewpoints is carried out in two steps. First, the untransformed point cloud of the robot arm at its home position is transformed to the pose in the image by using Pose Transformation matrices.

When a rotation is made on the n^{th} joint of the robot arm point cloud, the reference frame defined at that joint is rotated by the same angle of rotation as shown in Fig. 5. All the points that belong to the links that lie after the n^{th} joint are affected by this rotation. However they remain in the same coordinates with respect to n^{th} reference frame before and after the rotation, since the frame too has rotated.

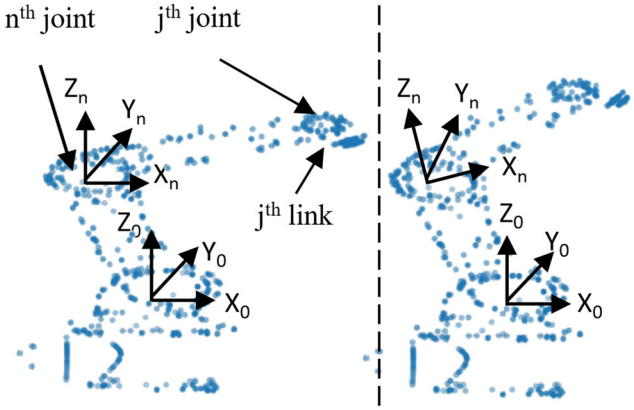


Fig. 5. Coordinate frames before and after transformation

Therefore, effectively, rotation applied to a point while keeping its reference frame fixed is equivalent to applying that same rotation to the reference frame while keeping the point fixed with respect to the frame. Therefore, the point cloud of a j^{th} link (which lies after the n^{th} joint) referred to a previous frame can be written as in equation (1)

$$^{n-1}[P_{link\ j}]_{new} = [^{n-1}T_n] \times {}^{n}[P_{link\ j}]_{initial} \quad (1)$$

Here $^{n-1}[P_{link\ j}]_{new}$ denotes the point cloud of the link at $(n\text{-}1)^{th}$ frame after applying the rotation to n^{th} joint. $^{n}[P_{link\ j}]_{initial}$ denotes the point cloud of the link at n^{th} frame before rotation. $[^{n-1}T_n]$ denotes the transformation matrix from $(n\text{-}1)^{th}$ reference frame to n^{th} reference frame after rotating the n^{th} frame by the relevant joint angle.

This can be expanded to refer the point cloud in its base frame, by multiplying by the rotation matrices corresponding to previous joint frames, as shown in equation (2)

$$^{0}[P_{link\ j}]_{new} = [^{0}T_1]..[^{n-1}T_n] \times {}^{n}[P_{link\ j}]_{initial} \quad (2)$$

The combined multiplication of all these transformation matrices in equation (2) can be considered as the Pose

Transformation Matrix corresponding to the nth joint ($[T_{P,n}]$), and it should be applied to all the points that lie after the nth joint. This procedure should be repeated six times to apply all the joint angles progressively from the 1st joint to the 6th joint. After all six angles are applied, the transformed point cloud will resemble the robot arm in the input image, and is referred to as the reference point cloud ($[P]_{ref}$).

Once the reference point cloud with the same pose as in the image is obtained, the second step is to transform that point cloud to different viewpoints by using Viewpoint Transformation matrices. Each viewpoint is defined as a defined rotation of a camera frame (that is at a fixed distance away from the base of the robot) in a spherical coordinate system having the robot arm base frame at its centre. The shift from robot's base frame to a viewpoint's camera frame is given by a transformation matrix, as in equation (3)

$$^{c1}[P]_{ref} = [^{c1}T_0] \times {}^{0}[P]_{ref} \quad (3)$$

Here, $^{c1}[P]_{ref}$ denotes the first viewpoint of the point cloud. $[^{c1}T_0]$ denotes the transformation matrix from the base frame to the first viewpoint's camera frame, also referred to as the first Viewpoint Transformation Matrix ($[T_{V,1}]$). This procedure is repeated until desired number of reference viewpoints are obtained. Viewpoints can be varied by changing the azimuth angle and polar angle to desired values. In this experiment, it was decided to use four viewpoints.

After the reference viewpoints are generated, they are compared with predicted viewpoints and the error is used to refine the predictions. This training is carried out until the error in the final output falls inside a small, fixed margin.

B. *Pose Estimator*

The output from the perspective generator contains robot arm point clouds at different viewpoints. The point clouds in these viewpoints are then transformed to one common viewpoint (same as the viewpoint of the untransformed point cloud) using inverse of viewpoint transformation matrices, and the average of the coordinate values are taken for each point to obtain predicted point cloud ($[P]_{ref}$).

Now, the untransformed point cloud at the home position, and the predicted point cloud at the desired pose are in the same viewpoint. Regardless of the poses, the origins of the 1st joint's reference frame in both point clouds will be at the same location, meaning that the displacement vector between those two frames is zero.

Therefore, it is possible to apply a rotation to the 1st joint of untransformed point cloud in such a way that its 1st link can coincide with the 1st link of predicted point cloud, as shown in equation (4)

$$^{1}[P_{link\ 1}]_{predict} = [R_{\theta 1}] \times {}^{1}[P_{link\ 1}]_{initial} \quad (4)$$

Here, $^{1}[P_{link\ 1}]_{predict}$ denotes the points belonging to 1st link in the predicted point cloud expressed in 1st joint's frame, $^{1}[P_{link\ 1}]_{initial}$ denotes the points belonging to 1st link in the untransformed point cloud expressed in 1st joint's frame.

Since $^{1}[P_{link\ 1}]_{predict}$ and $^{1}[P_{link\ 1}]_{initial}$ are known, the unknown variable $[R_{\theta 1}]$ can be estimated using linear regression, and it results in a 3 × 3 rotation matrix. This corresponds to the rotation that should be applied to 1st joint, for the points of the 1st link in untransformed point cloud to coincide with the points of the 1st link in predicted point cloud. The first joint angle (θ_1) can then be estimated by comparing each element of the generated rotation matrix to a standard rotation matrix expressed as a mathematical function.

After θ_1 is estimated, -(θ_1) is applied to the predicted point cloud, to eliminate the effect of rotation of the 1st joint from the rest of the robot. Now the origins of 2nd joint's reference frame in both point clouds will be at the same location. By repeating the same procedure, all six joint angles can be estimated through linear regression.

C. *Optimizer*

If a part of the robot arm is not visible in the image, then the 3D prediction corresponding to that part can include false points and may not represent the pose correctly. Therefore, the falsely represented data points have to be eliminated from the predicted point cloud for better estimation of joint angles.

In order to eliminate the falsely predicted points, the untransformed point cloud at home position is transformed with the predicted set of joint angles by using pose transformation matrices. This new, transformed point cloud and predicted point cloud are then superimposed and the absolute displacement error for each point in both point clouds is calculated. The point with the highest value of error is considered a false prediction and is therefore eliminated. Joint angles are then estimated again by linear regression similar to the pose estimator component. This process is repeated until the maximum error falls below a predefined value, eliminating the false predictions as much as possible, as shown in Fig. 6. This results in an optimized set of joint angles which is the final output from the system.

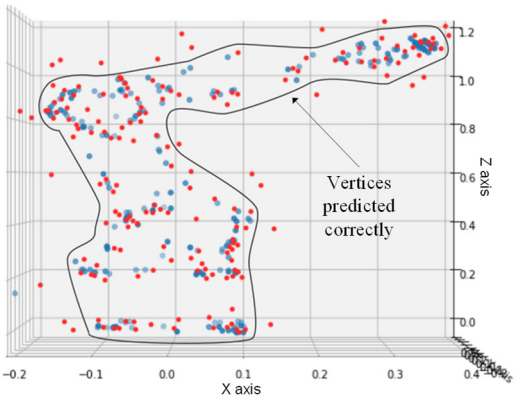


Fig. 6. Representation of predicted and transformed vertices.

A disadvantage in estimation of pose by regression is that the errors in the joint angles will get accumulated when joint angles are estimated progressively from the 1st joint. However, the main reason for falsely predicted vertices is poor visibility and the chances of the first few joints being poorly visible are quite low compared to the latter joints. Therefore, the effect of error accumulation will not be significant.

IV. Data Generation

Two sets of data are required to implement the proposed approach: RGB images of the robot arm poses and the 3D representation with surface points of the corresponding pose in multiple viewpoints.

The RGB images of poses are generated using Blender's EEVEE render engine. To minimize the gap between real and synthetic data, domain randomization techniques were employed, including random placement of distractor objects in the background, variation of colour intensity, variation of texture of wall and floor and variation of number of lights in the scene and their position [9].

The point cloud representation of the robot arm consists of significantly high number of vertices. The computational cost of training a neural network to predict all the vertices would be very high. Vertices that are closer to each other would have less significance in predicting the pose. Therefore, the number of vertices have to be reduced to make the model computationally efficient and accurate. Reduction in the number of vertices is carried out in two steps: Pearson's Correlation Coefficient and Wrapper Feature Selection.

Pearson's correlation coefficient is a measure of linear correlation between two sets of data. A database of vertex coordinates for different poses is created and the dependency between each vertex point is calculated using the Pearson's coefficient formula and presented as a matrix. Higher adjacency values between two vertices indicate that they are highly dependent on each other and are therefore eliminated.

These reduced vertices are then passed through a wrapper filter-based feature selection algorithm. This calculates the individual effect of all the vertices in determining the overall outcome. Vertices that have less significance in estimating the pose are then eliminated. The combined effect of Pearson's correlation coefficient and wrapper filter reduces the number of feature vertices significantly (nearly 100 times) but still retains the core shape necessary for pose estimation as illustrated in Fig. 8.

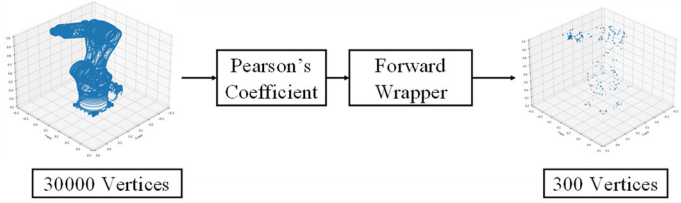
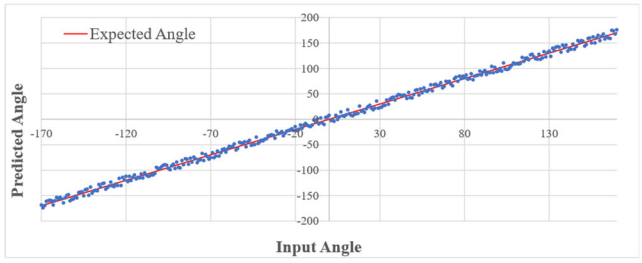


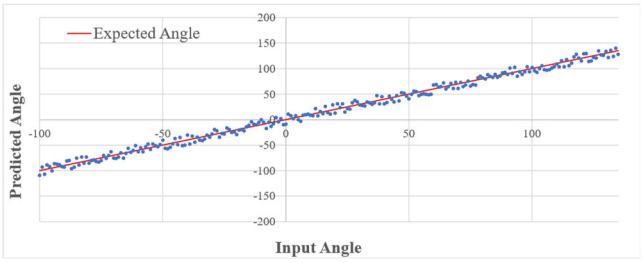
Fig. 8. Combined process for reduction of vertices

V. Evaluation

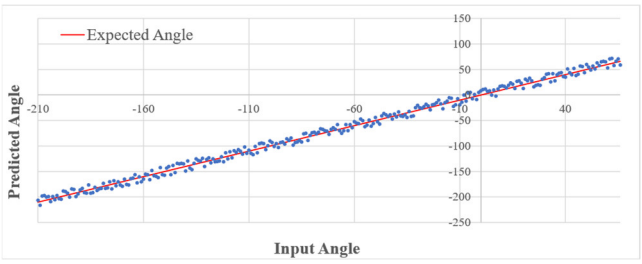
To determine accuracy of prediction of the joint angles independently, each joint angle is allowed to iterate through a range of values with an increment of 1 degree, while the other joint angles are randomly selected. At every iteration, the



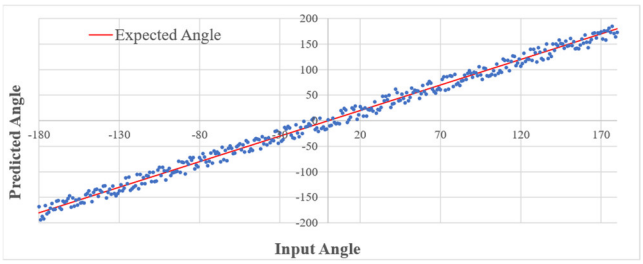
(a)



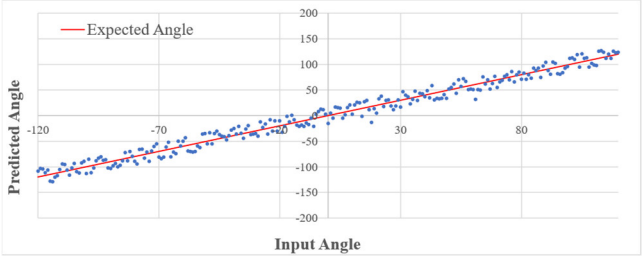
(b)



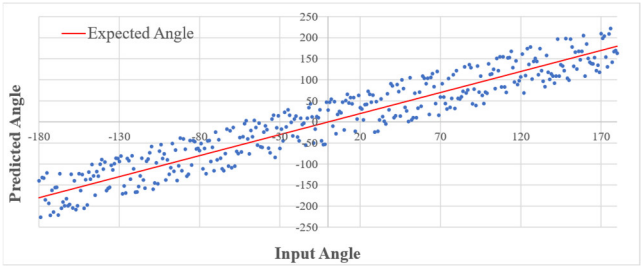
(c)



(d)



(e)



(f)

Fig. 9 Comparison of the predicted angles and expected angles (same as input) for a) θ_1 b) θ_2 c) θ_3 d) θ_4 e) θ_5 f) θ_6

values and corresponding pose image are input to the framework and pose is estimated. The accuracy of the model is defined as the root mean squared error (RMSE) between the predicted joint angle and the actual joint angle input to the model. Fig. 9 illustrates predicted angles for each joint angle of the robot arm. They are compared against the expected angles (which should be equal to the commanded input value), shown as straight line of gradient 1.

As we move from the 1st joint to the 6th, the portion of the robot arm that uniquely reflect a change in that joint angle decreases. Hence the accuracy of prediction also decreases, as shown in the graphs of Fig. 9. A change in θ_6, is uniquely reflected only by the end effector. Since the end effector used is very small compared to the rest of the robot arm body, it contributes to a higher error for prediction of θ_6.

A drawback in calculating accuracy is that it can vary significantly depending on the pose of the robot arm and the position and orientation of the camera. If a certain portion of the robot arm is not visible in the image taken by the camera, then the 3-dimensional predictions corresponding to that part will be highly inaccurate and would result in inaccurate pose angles. Even though this error is reduced by the optimizer described in III C, if a major portion of a link is not visible, then the error would still be high. Therefore, to analyse this effect better, accuracy values are calculated for poses with different levels (approximate) of link visibility and are presented in Table II.

TABLE II
RMSE FOR DIFFERENT VISIBILITIES OF LINKS

Link	Visibility (%)	RMSE
1	30	6.7
	50	2.7
	70	2.2
2	30	10.2
	50	3.9
	70	3.1
3	30	12.2
	50	6.8
	70	6.1
4	30	41.2
	50	16.1
	70	7.4
5	30	34.1
	50	10.1
	70	6.1
6	30	88.7
	50	56.7
	70	22.1

In the pose estimator component of the model, the errors will get accumulated, and therefore the latter joint angles would have higher errors. However, it can be observed from Table II that θ_5 has lower error than θ_4. This is due to the optimizer module which finds the joint angle from only the points that correctly represent the 3D robot arm. Therefore, even though θ_5 predicted by the pose estimator initially had higher error due to accumulated error from θ_4, it has been reduced by the optimizer as the visibility of the 5th link is better.

VI. CONCLUSION

This paper proposes an approach to determine the pose of a robot arm manipulator using computer vision and deep learning, where a single RGB camera is fixed at a distance from the manipulator. The novelty of this approach is that it employs deep learning to predict the 3D surface of the robot arm manipulator from the 2D image without any depth information, and this 3D surface is used to predict the pose. It also makes use of optimizer algorithms to keep the predicted surface points as minimal as possible without affecting the final accuracy of pose prediction. The manipulator is modelled virtually with its kinematics and is used to construct the 3D projections associated with the input 2D image. This study uses 'KUKA KR 6 R900 sixx' robot as an illustrative example. However, this approach can be applied to any robot arm by following the same procedure. This approach of pose estimation does not entirely replace the function of encoders, rather, it can be used as a reference solution when the encoders are malfunctioning. The experimental results demonstrate the feasibility of the proposed approach, and the ability to predict for poses where the visibility of one or more links are affected.

ACKNOWLEDGEMENT

Authors gracefully acknowledge the financial support extended by the Senate research committee of the University of Moratuwa Sri Lanka.

REFERENCES

[1] X. Gratal, J. Bohg, M. Björkman, D. Kragic, "Scene representation and object grasping using active vision," IROS 2010 Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics, 2010.

[2] F. Zhou, Z. Chi, C. Zhuang, H. Ding, "3D Pose Estimation of Robot Arm with RGB Images Based on Deep Learning," 2019 International Conference on Intelligent Robotics and Applications (ICIRA), 2019, pp. 541-553, doi: 10.1007/978-3-030-27538-9_46.

[3] J. Bohg, J. Romero, A. Herzog and S. Schaal, "Robot arm pose estimation through pixel-wise part classification," 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 3143-3150, doi: 10.1109/ICRA.2014.6907311

[4] Y. Kuo, B. Liu and C. Wu, "Pose Determination of a Robot Manipulator Based on Monocular Vision," in IEEE Access, vol. 4, pp. 8454-8464, 2016, doi: 10.1109/ACCESS.2016.2633378.

[5] F. Widmaier, D. Kappler, S. Schaal and J. Bohg, "Robot arm pose estimation by pixel-wise regression of joint angles," 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 616-623, doi: 10.1109/ICRA.2016.7487185.

[6] J. Miseikis, I. Brijacak, S. Yahyanejad, K. Glette, O. J. Elle and J. Torresen, "Multi-Objective Convolutional Neural Networks for Robot Localisation and 3D Position Estimation in 2D Camera Images," 2018 15th International Conference on Ubiquitous Robots (UR), 2018, pp. 597-603, doi: 10.1109/URAI.2018.8441813.

[7] C.H. Lin, C. Kong, and S. Lucey. 2017. "Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction." ArXiv Preprint ArXiv:1706.07036.

[8] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, S. Birchfield, "Camera-to-Robot Pose Estimation from a Single Image," 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 9426-9432.

[9] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 23-30.