End-To-End Real-Time Visual Perception Framework for Construction Automation

Mohit Vohra[1], Ashish Kumar[1], Ravi Prakash[1] and Laxmidhar Behera[1,2], *Senior Member, IEEE*

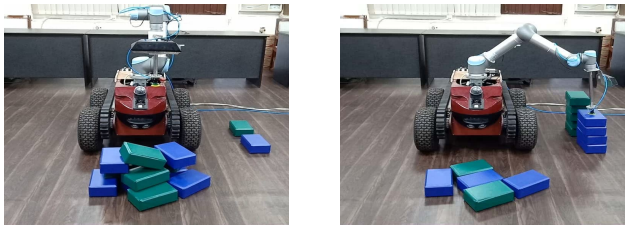
***Abstract*— In this work, we present a robotic solution to automate the task of wall construction. To that end, we present an end-to-end visual perception framework that can quickly detect and localize bricks in a clutter. Further, we present a light computational method of brick pose estimation that incorporates the above information. The proposed detection network predicts a rotated box compared to YOLO and SSD, thereby maximizing the object's region in the predicted box regions. In addition, precision (*P*), recall (*R*), and mean-average-precision (mAP) scores are reported to evaluate the proposed framework. We observed that for our task, the proposed scheme outperforms the upright bounding box detectors. Further, we deploy the proposed visual perception framework on a robotic system endowed with a UR5 robot manipulator and demonstrate that the system can successfully replicate a simplified version of the wall-building task in an autonomous mode.**

I. Introduction

Manufacturing and construction are one of the widespread and continuously growing industries. The former has seen a dramatic increase in production capacity due to the optimization of industrial automation, while the latter has adopted automation only marginally [1]. Construction automation is inherently quite challenging for several reasons. First, the workspace is highly unstructured. Therefore, very high precision and robust visual perception, motion planning, and navigation algorithms are required for autonomous solutions to adapt to different scenarios. Secondly, a mobile manipulator needs to move between multiple positions, compelling us to perform onboard computations for various algorithms. Therefore, limited memory, power, and computational resources make this task more challenging.

The process of automation can have a broad impact on the construction industry. First, construction work can continue without pause, which ultimately shortens the construction period and increases economic benefits. Also, the essential benefits of construction automation are worker safety, quality, and job continuity. Towards this end, recently, Construction Robotics, a New York-based company, has developed a bricklaying robot called SAM100 (semi-automated mason) [2], which makes a wall six times faster than a human. However, their robot required a systematic stack of bricks at regular intervals, making this system semi-autonomous, as the name suggests.

One of the primary construction tasks is to build a wall from a pile of randomly arranged bricks. To replicate the simplified version of the above work, humans must complete a sequence of operations: *i)* Select the appropriate brick from the pile, e.g., the topmost brick, *ii)* finding the optimal grasp pose for the brick, and *iii)* finally, placing the brick in its desired place, i.e., on the wall. Humans can do this work very quickly and efficiently. However, the robot must perform a complex set of underlying operations to complete the above steps autonomously [3].



(a) (b)

Fig. 1: (a) shows a simple scenario where a pile is located near the robotic system, (b) the robotic system mimics wall building task i.e. detects pile, selects a target brick and constructs a wall on its side in a fully autonomous way.

In this paper, we aim to deploy a robotic solution for the task of construction automation in a constrained environment (Fig.1) with limited computational resources (single CPU with I7 processor, 4core, 8GB RAM). We assume that all bricks are of equal size, and their dimensions are known. We further assume that the wall assembly area and brick piles are very close, exempting us from deploying any localization and navigation modules for robots. Thus, the main challenge in this task is to detect and localize bricks in the clutter while handling the multiple instances of the bricks. Once we have localized the bricks in a clutter, we will use the above information to estimate the brick pose. Following are the main contributions in this paper:

- A computationally efficient object detection network for the detection and localization of bricks in a clutter is presented in this paper.
- A light computational method for estimating brick pose using point cloud data is presented in this paper.
- All the modules are integrated into a robotic system to develop a fully autonomous system.
- Extensive experiments to validate the performance of our system.

In the next section, we briefly provide a review of state-of-the-art algorithms related to the paper. In the section-III, we formulate the problem statement. The overall approach and its modules are explained in section-IV. In section-V, the experimental study of the algorithm is reported for various

[1]Authors are with Indian Institute of Technology Kanpur, India, {mvohra, krashish, ravipr and lbehera}@iitk.ac.in.
[2] Author is with TCS Innovation Labs, Noida, India

test cases. This paper is finally concluded in section-VI.

II. Related Works

A. Object Detection

As mentioned in the previous section, the first stage of the construction process is the localization of the target object. In our case, the target object is referred to as a brick. In general, bricks are arranged randomly. Therefore, the brick must be localized before grasping. The process of brick localization falls under the category of the object detection algorithm. We perform a brick localization process in the image space. Several object detection algorithms exist in the literature. Here, we limit our discussion to only Conventional Neural Network (CNN) based methods.

The RCNN [4] generates object proposals (rectangular regions) in the image plane, and a CNN is used to extract features from the proposed regions, followed by a classifier to classify the proposed regions into N different classes, where N may vary according to application. In RCNN, most of the time is consumed in proposal generation as this step is performed on CPU, and also inference time increases linearly with an increase in the number of proposals. SPPnets [5] were proposed to speed up the RCNN by extracting the features for the whole image at once and then cropping the feature map corresponding to the proposals. Due to the multistage nature of the above algorithm, joint training was required. Fast-RCNN [6] proposes an improved approach that is relatively faster and requires single-stage training. A further improved version of Faster-RCNN [7] was also proposed in which proposals are generated within the CNN, called a Region Proposal Network (RPN). The RPN was the key improvement in improving the overall algorithmic real-time performance. All the methods discussed above predict an upright bounding box around the detected object. In addition to the target object region, the predicted box may contain non-object regions or backgrounds. Hence, to minimize the background in the detected boxes, various solutions are present in the literature. For example, in [8], the author predicts a rotated bounding box from the set of prior rotated boxes (anchor boxes). Similarly, Mask-RCNN [9] can predict the bounding box and mask of the object simultaneously, which is known as instance detection and segmentation.

All the algorithms mentioned above consists of two steps; *i)* generation of object proposals or anchor boxes (axis-aligned or rotated), *ii)* classification (or regressing) the proposals using a CNN with a backbone such as VGG [10], ResNet [11]. Thus the performance of the algorithm depends on the proposal generation process. On the other hand, authors of You Only Look Once (YOLO-v1) [12] have proposed a single network for object detection which divides the image into grid cells and directly predicts the fixed number of bounding boxes, corresponding confidence scores, and class probabilities for each grid cell. In the same direction, single shot multibox detector (SSD) [13] is another variant of a single-stage object detector. In this variant, multi-resolution object detection is performed, i.e., detecting the presence of an object and its class score at various stages of different spatial resolutions.

B. 6D Pose Estimation

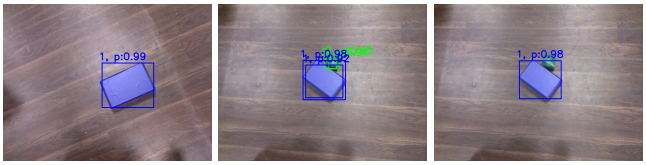
After brick localization, a grasp operation needs to be performed by the manipulator. Choosing an optimal grasp configuration is a non-trivial task and remains an open problem. The grasp configuration depends on the 6D pose of the brick. Several Neural-network based pose estimation methods exist in the literature [14], [15], but limited memory resources compel us to use computationally light pose-estimation methods. To this end, several algorithms exist to estimate the object's 6D poses, which require a high degree of surface texture on the object. In our case, the estimation of brick poses is quite challenging due to their cubic shape (flat surfaces), which do not have surface textures. Therefore, the feature point matching technique [16] [17] cannot be used. Other approaches [18], [19] consider an earlier model of the object. These methods require a preprocessing step, followed by the correspondence matching process. The matching process is the most time-consuming component of such algorithms. Besides, point-to-point matching methods (ICP [20], GICP [21]) are based on local geometric properties. Therefore, these methods can be stuck in local minima when aligning the target model with the reference due to flat surfaces.

III. Problem Statement

A. Object Detection

As mentioned in previous sections, the main challenge is to identify the bricks in a clutter. All state-of-the-art object detectors predict the upright or straight bounding box, which has three limitations:

- The bounding box corresponding to the rotated or tilted object contains a significant non-object region (Fig. 2a). Thus, it requires an additional step to extract object information, like object segmentation in the bounding box region.
- If two or more objects are very close to each other. The corresponding bounding boxes will have non-zero intersecting regions (Fig. 2b). Thus, additional steps are required to handle intersecting regions, as this region may contain clutter for one box or an object part for another box.
- If the intersecting regions are significant, after applying non-maximal-suppression (NMS), neighbor detection may be missed [22], as shown in Fig. 2c.



(a) Predicted Boxes (b) Boxes Overlap (c) After NMS

Fig. 2: Nearby predictions can be missed due to NMS

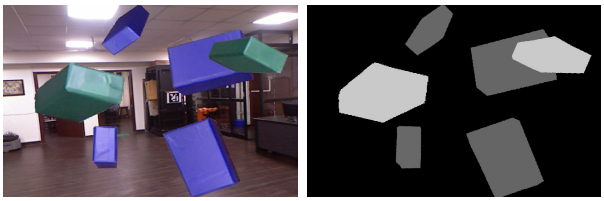


Fig. 3: A synthetically generated sample

To compete with the above limitations, we designed a CNN-based model to detect and localize bricks by predicting rotated boxes. An additional degree of freedom, i.e., the box's angle, allows the network to predict the box with greater alignment with the target object. Since most of the area inside the rotated bounding box corresponds to the target object, we can directly use the region corresponding to the rotated bounding box to extract the target object's information, avoiding any additional computations. A detailed description of the proposed model is given in Section-IV.

B. 6D Pose Estimation

As mentioned earlier, the bricks used in the experiments have a flat and textureless surface. Therefore, feature matching methods for pose estimation are unreliable in our experiments, as the number of features is less and not very distinct. Since the bricks used in our experiments have a cuboidal shape, if we can estimate the pose of at least one surface of the brick, this information is sufficient to estimate the pose of the entire brick. The brick has six faces, and each face has a specific relative pose with a local brick frame. Hence to estimate the brick pose, we have to identify the brick surface (out of six surfaces), estimate the surface pose, and use relative transformations to get the complete brick pose. A brief description of the pose estimation method is given in the section - IV-C.

IV. The Proposed Framework

A. Dataset

We collect 30 images for each class of brick (e.g., blue and green) using Kinect. Images are collected, such that each image has an isolated brick with a different pose. A manual mask is generated for each image. Following [23], the raw images and the corresponding masks are used to synthesize the cluttered scenes. A dataset of 10k training images and 5k test images are generated. We generate ground truths for synthetic images such that a unique instance-ID, as opposed to semantic segmentation, is assigned for each brick instance. A random sample from the data set is shown in Fig. 3. Furthermore, for each mask instance, we generate a rotated box using the OpenCV API.

Each image is divided into several grids, where each grid has a size of 16×16 pixels. Thus if the raw image has a size of 480×640, then the total number of the grids are $\frac{480}{16} \times \frac{640}{16}$, i.e., 30×40. Further, each grid is represented by an $8D$ vector representing the three-class probabilities (blue brick, green brick, or background) and five bounding box parameters x, y, w, h, θ. For each grid, class probabilities are assigned if the rotated bounding box's centroid (corresponding to the blue brick, green brick) exists within that grid. If there is no centroid in the grid, then we will assign the probability of 1.0 to the background label. Suppose a centroid exists within a grid. Corresponding bounding box parameters are x, y, w, h, θ, where x, y is the offset between the rotated bounding box center and the topmost grid corner. Parameters w, h, θ are width, height, and orientation of the bounding box, respectively. We scale the bounding box parameters in the range of $(0, 1)$, where the maximum value of offset is 16 pixels. The box's maximum dimension can be 480×640 pixels, and the maximum orientation value is 3.14 radians. Thus for each image, we have an output tensor of size $30 \times 40 \times 8$. Further, if multiple centroid points exist in a single grid, we select the centroid point corresponding to the mask, which has a larger fraction of the complete mask in that grid.

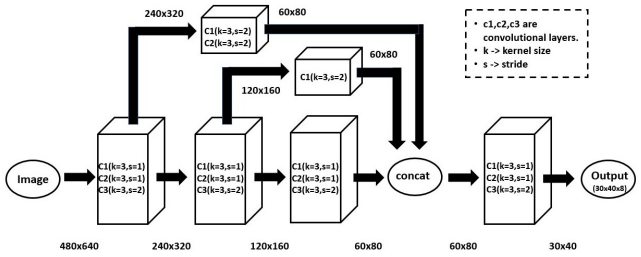


Fig. 4: Proposed Network

B. Rotating Box Network

Fig. 4 represents the network architecture. For each block, the size of the input and output feature map is mentioned. Also, we use the ReLU activation function after each layer. The SSD architecture inspires the proposed rotating box network architecture. In SSD, shallow layer features and depth layer features are used for final predictions. Similarly, in the proposed network, features from individual shallow layers are processed, concatenated, and passed through a series of fully convolutional layers for final prediction. Unlike SSDs, the proposed network prediction does not use any anchor boxes. Instead, it predicts an additional degree of freedom (angle of the box), and thus the predicted bounding boxes can align more accurately than the constrained bounding box.

In order to train the network for predicting rotated boxes, the input to the network is the raw image, and the output of the network is a tensor of size $30 \times 40 \times 8$. Further, we use a cross-entropy loss for the class probabilities and a regression loss for the bounding box parameters. Overall loss for the network is the average of both losses. Further, to avoid any biasing in training because of the large number of non-object grids as compared to the object grids, we select positive to negative ratio = $1 : 2$ by following [13]. Output the model for the different arrangement of bricks in a variety of backgrounds is shown in Fig. 5.

C. Pose Estimation

To estimate the brick pose, as mentioned earlier, we have to calculate the pose of one of the brick surfaces and use

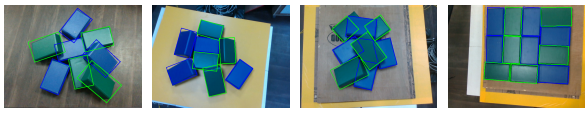
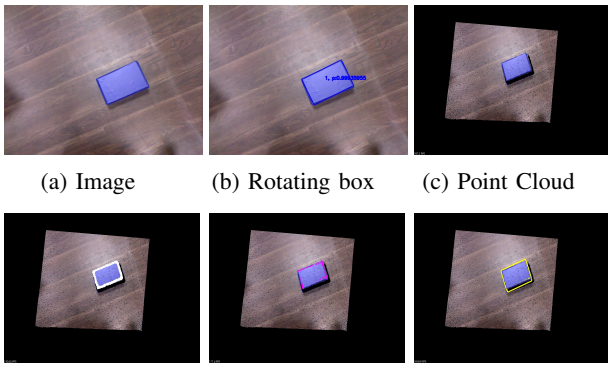


Fig. 5: Network Predictions

relative transformation to extract the complete brick pose. For this task, we feed the current image (Fig. 6a) to the rotating box network. The region corresponding to the rotating box (Fig. 6b) is called the brick region, and the point cloud corresponding to the brick region is called the brick cloud. On the brick's cloud, we apply the following steps:

- Apply *RANSAC* method for estimating a set of points (inliers) that fits a planar surface in the brick cloud data.
- Compute the centroid, major axis, and minor axis of the inliers. Together these three pieces of information represent the pose of the planar surface. To estimate the surface ID, we follow the following steps.
- Using [24], extract all boundary points in the inliers, which is marked in white color in Fig. 6d.
- Apply RANSAC method for fitting the lines on the boundary points which are shown pink in Fig. 6e.
- Compute all corner points, which are the intersecting point of two or more lines.
- Pair the corner points representing the same line [25], and the distance between two corner points gives the length of the edge.
- Since the brick dimensions are known in advance. Hence the length of the edges can be used to identify the surface, and we can use relative transformation to compute the 6D pose of the brick, as shown in Fig. 6f.



(a) Image (b) Rotating box (c) Point Cloud

(d) Edges (e) Lines (f) 6D Pose

Fig. 6: Pose estimation pipeline

V. Experiments and Result

A. Experimental Setup

For experimental evaluation, we use our robotic platform setup, as shown in Fig. 1a. It consists of a UR5 robot manipulator with its controller box (internal computer) mounted on a ROBOTNIK Guardian mobile base and a host PC (external computer). The UR5 robot manipulator is a 6-DOF robotic arm designed to work safely alongside humans. We use an eye-in-hand approach, i.e., the image acquisition hardware, which consists of RGB-D Microsoft Kinect Sensor, is mounted on the manipulator. A suction-based gripper is used for grasping. Robot Operating System (ROS) is used to establish a communication link among the sensor, manipulator, and the gripper.

B. Overall Algorithmic Flow

For the experiment, we follow a simple pattern (or wall pattern) such that we place a blue brick on the previously placed blue brick and a green brick over the previously placed green brick and keep on placing the bricks up to the height of 6 layers. The system needs to place the brick correctly for wall construction, hence requiring the brick pose with high accuracy. Since in a dense clutter, network prediction for brick can include some portion of other bricks, directly processing the box regions for pose can give a noisy or less reliable brick pose. For the safer side, we perform the grasp operation with a noisy brick pose and place the brick in a separate area and again estimate the pose of single isolated brick.

Fig. 7 refers to the sequence of steps executed to complete a single phase of the wall construction task. In the first stage, the Kinect sensor is positioned to have a clear view of the bricks clutter (Fig. 7a), and the current image is fed to the rotating bounding box network. We compute the planar surface and its pose (centroid, major-axis, and minor-axis) corresponding to each rotated box. We select the topmost pose from all the calculated poses, i.e., the pose at the top of the clutter. The required motion commands are generated to reach the selected pose for brick grasping in clutter (Fig. 7b).

The grasped brick is placed in a separate area (Fig. 7c), and again the vision sensor is positioned in an appropriate position to have a clear view of a single isolated brick (Fig. 7d). At this instant, the current frame is fed to the network, and the output of the network is a class label and the bounding box parameters. The brick's pose is estimated and required motion commands are generated to grasp the brick (Fig. 7e).

In the final stage, the sensor is positioned in an appropriate position to have a view of the wall region (Fig. 7f). The current frame is fed to the network. Using the estimated bounding box, point cloud data, and grasped brick label (previous step), the brick's final pose is estimated, where the brick is to be placed to form a wall (Fig. 7g).

C. Error Metrics

The system's overall wall-building performance depends entirely upon the performance of the visual perception system, i.e., the accuracy of brick detection. Therefore, we report the performance of the detection system in terms of precision (P) and recall (R). We have defined P, C as:

$$P = \frac{NO}{TP}, \quad R = \frac{DB}{TB} \tag{1}$$

where,

NO Number of object pixels in the predicted box (rotated / upright)

(a) (b) (c) (d) (e) (f) (g)

Fig. 7: Sequence of actions executed in order to carry out a single step of wall construction

TP Total number of object pixels in the predicted box (rotated / upright)
DB Total number of detected bricks
TB Total number of the bricks in the ground truth.

D. Quantitative Analysis

We compare rotating box network with YOLO-v3 and SSD-lite. For fair comparison, all models are trained by choosing hyper-parameters as $epoch = 60, mini - batch = 4, lr = 0.0001$ except SSD-lite which has a learning rate of 0.001. We use Adam optimizer to tune the parameters of CNN. We divide the quantitative analysis in following two cases:

1) Upright Bounding Box Prediction: In this case, we report Mean Average Precision (mAP) score, P and R of the proposed CNN model against SSD-lite and YOLO-v3 (Table-I). All the three models produces upright bounding boxes.

TABLE I

	SSD-lite	YOLO-v3	Proposed
P	0.608	0.580	0.638
R	0.98	0.84	0.84
mAP	0.834	0.827	0.811

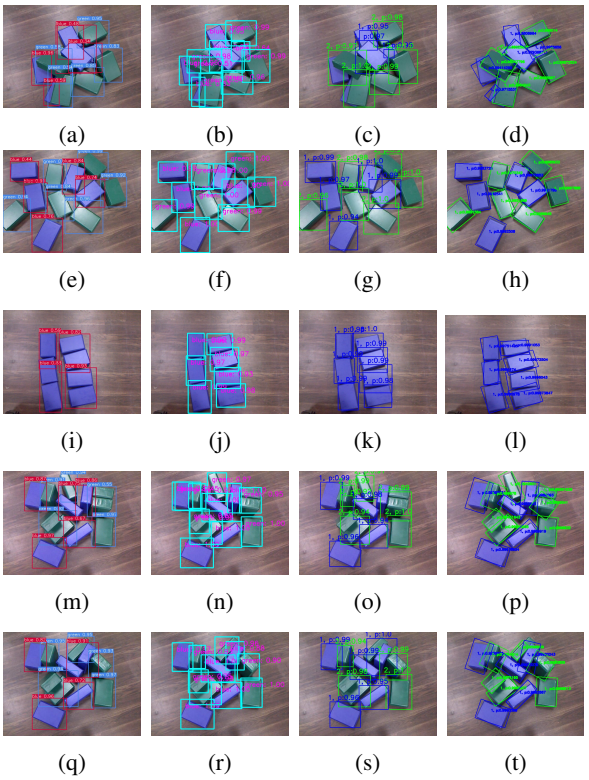
2) Rotated Bounding Box Prediction: In this case, SSD-lite and YOLO-v3 produce regular bounding boxes, while the proposed CNN model produces rotated boxes. Since the mAP score of rotated boxes can not be compared directly with that of upright bounding boxes. Hence only P and R is reported (Table-II). The precision P for rotating boxes network is significantly higher as compared to other networks, because of the one additional degree of freedom (angle of boxes), the network predicts bounding boxes that can align more accurately as compared to constrained bounding boxes (straight boxes), thus there will be less overlap between different boxes and most of the region inside the bounding box represents the same object which results in high precision.

TABLE II

	SSD-lite	YOLO-v3	Proposed
P	0.608	0.580	0.778
R	0.98	0.84	0.999

E. Qualitative Analysis

Predictions of all four networks under consideration are shown in Fig. 8. It can be noticed from Fig. 8m and 8o, two green bricks present in the center of the frame are represented by a single bounding box, thus decreasing the recall value for Yolo-v3 and the proposed bounding box network. While SSD-lite (Fig. 8n) and the proposed rotating box network



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m) (n) (o) (p) (q) (r) (s) (t)

Fig. 8: From Column 1 to 4: YOLO-v3 predictions, SSD-lite predictions, Proposed Bounding box predictions, Rotating box predictions.

(Fig. 8p), both assign two different boxes for the two green bricks. Thus SSD-lite and our network have a higher recall. However, in SSD-lite, two predicted bounding boxes have a significant overlap area, thus having lower precision than the rotating box network.

F. Task Evaluation

To evaluate the system's performance, we repeat the task of wall construction for 25 rounds. For each round, the robotic system has to place the bricks up to the height of 6 layers, and the wall pattern will remain the same, i.e., the blue brick on previously placed blue brick and green brick on previously placed green brick. For the experiments, the first layer of bricks is placed manually, and the system has to place the rest of the bricks, i.e., $2 - 6$ layers, according to the pattern. We count the number of bricks (or layers) for each round the robotic system has successfully placed on the wall. In the experiments, we define the brick placement as successful if the distance between the centroid of the

currently placed brick and the centroid of the manually placed brick, when projected on the ground plane is $< 0.1m$. Further, the Euler angle difference between the calculated pose of the currently placed brick and the manually placed brick should be less than $15°$ for each axis. From our experiments, we observed that if none of the above criteria are satisfied, the wall becomes asymmetrical and collapses. The Table-III shows the performance of the system for 25 rounds. The video link for the experiment is https://www.youtube.com/watch?v=FvsCv-Pt58c.

TABLE III: Task Evaluation

	layer-2	layer-3	layer-4	layer-5	layer-6
Successful rounds (max 25)	25	25	22	19	17

From Table-III, we observed that the robotic system has successfully placed layer-2 and layer-3 bricks in all 25 rounds. However, accuracy decreases with the upper layers. This is because the new position of the brick on the wall is estimated by calculating the previously placed brick pose. Thus, if there is a slight error in brick placement in the previous step, this error is transferred to the current step. Thus, with higher layers, the error accumulates, resulting in lower accuracy.

VI. Conclusion

An end-to-end visual perception framework is proposed. The framework consists of a CNN for predicting a rotated bounding box. The performance of the CNN detector has been demonstrated in various scenarios, which mainly include isolated and dense clutter of bricks. The proposed CNN module localizes the bricks in a clutter while simultaneously handling multiple instances of the bricks. The detection is free of the anchor-box technique, which improves the timing performance of the detection module. In order to compare our method quantitatively with state-of-the-art models, we reported Precision (P), Recall (R), and mAP scores for various test cases. We have compared the effectiveness of rotating bounding box predictions against upright bounding box detection (YOLO-v3, SSD-Lite). The proposed scheme outperforms the upright bounding box detection. It implies that rotating bounding boxes can align more accurately with the object's convex-hull and thereby reduce the overlap with neighboring bounding boxes(if any). The framework has also been successfully deployed on a robotic system to construct a wall from bricks in fully autonomous operation.

References

[1] K. Asadi and K. Han, "Real-time image-to-bim registration using perspective alignment for automated construction monitoring," in *Construction Research Congress*, vol. 2018, pp. 388–397, 2018.

[2] S. Parkes, "Automated brick laying system and method of use thereof," Jan. 31 2019. US Patent App. 16/047,143.

[3] R. Prakash, M. Vohra, and L. Behera, "Learning optimal parameterized policy for high level strategies in a game setting," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1–6, IEEE, 2019.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[6] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[8] S. Li, Z. Zhang, B. Li, and C. Li, "Multiscale rotated bounding box-based deep learning method for detecting ship targets in remote sensing images," *Sensors*, vol. 18, no. 8, p. 2702, 2018.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[14] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 292–301, 2018.

[15] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11632–11641, 2020.

[16] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2155–2162, IEEE, 2010.

[17] J. P. S. do Monte Lima and V. Teichrieb, "An efficient global point cloud descriptor for object recognition and pose estimation," in *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 56–63, IEEE, 2016.

[18] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 998–1005, Ieee, 2010.

[19] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige, "Going further with point pair features," in *European conference on computer vision*, pp. 834–848, Springer, 2016.

[20] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586–606, International Society for Optics and Photonics, 1992.

[21] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp.," in *Robotics: science and systems*, vol. 2, p. 435, Seattle, WA, 2009.

[22] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-nms–improving object detection with one line of code," in *Proceedings of the IEEE international conference on computer vision*, pp. 5561–5569, 2017.

[23] A. Kumar and L. Behera, "Semi supervised deep quick instance detection and segmentation," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8325–8331, IEEE, 2019.

[24] M. Vohra, R. Prakash, and L. Behera, "Real-time grasp pose estimation for novel objects in densely cluttered environment," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1–6, IEEE, 2019.

[25] M. Vohra., R. Prakash., and L. Behera., "Edge and corner detection in unorganized point clouds for robotic pick and place applications," in *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics - ICINCO,*, pp. 245–253, INSTICC, SciTePress, 2021.