3D Hand and Object Pose Estimation for Real-time Human-robot Interaction

Chaitanya Bandi[a], Hannes Kisner and Urike Thomas
Robotics and Human-Machine Interaction Lab.,
Technical University of Chemnitz, Reichenhainer str.70, Chemnitz, Germany

Keywords: Pose, Keypoints, Hand, Object.

Abstract: Estimating 3D hand pose and object pose in real-time is essential for human-robot interaction scenarios like handover of objects. Particularly in handover scenarios, many challenges need to be faced such as mutual hand-object occlusions and the inference speed to enhance the reactiveness of robots. In this paper, we present an approach to estimate 3D hand pose and object pose in real-time using a low-cost consumer RGB-D camera for human-robot interaction scenarios. We propose a cascade of networks strategy to regress 2D and 3D pose features. The first network detects the objects and hands in images. The second network is an end-to-end model with independent weights to regress 2D keypoints of hands joints and object corners, followed by a 3D wrist centric hand and object pose regression using a novel residual graph regression network and finally a perspective-n-point approach to solve 6D pose of detected objects in hand. To train and evaluate our model, we also propose a small-scale 3D hand pose dataset with a new semi-automated annotation approach using a robot arm and demonstrate the generalizability of our model on the state-of-the-art benchmarks.

1 INTRODUCTION

Hand and object pose estimation is an active research field for applications like robotics, augmented reality, and manipulation. 3D hand pose estimation and 6D object pose estimation have been addressed independently. Nevertheless, combined hand and object pose estimation enhances the mutual occlusions and this is yet to be solved for real-time applications. In this work, we aim to introduce a pipeline to estimate both hand pose and object pose for interaction scenarios. In robotics applications like bidirectional handover of objects, reactiveness, reliability, and safety are highly significant. This can be achieved by precise estimation of fingertips and object pose in real-time. The state-of-the-art works rely heavily on deep learning architectures for 3D hand pose estimation (Zimmermann and Brox, 2017; Mueller et al., 2018; Iqbal et al., 2018; Ge et al., 2019), 6D object pose estimation (Tekin et al., 2017; Peng et al., 2018; Li et al., 2018; Wang et al., 2019; Park et al., 2019; Labbé et al., 2020; Tremblay et al., 2018), and unified hand and object pose estimation (Doosti et al., 2020; Hasson et al., 2019; Hasson et al., 2020; Tekin et al., 2019).

The works (Yang et al., 2020; Rosenberger et al., 2020) propose unique solutions for applications like the handover of objects. These works rely on segmentation networks to obtain the region of hands and objects and later forward the region to the respective grasp pose refinement model. Although the segmentation networks are reliable, the inference speed is slow without proper hardware resources.

In this paper, we present an approach to regress 3D hand pose using deep learning architecture and compute 6D object pose estimation as illustrated in Figure 1. The first network is an independent object detection model to recognize the region of hands and objects. The second network consists of two different deep learning models that can either be trained end-to-end or independently to infer 2D hand pose, 3D hand pose, 2D object corners, 3D object corners and a perspective-n-point solver for 6D object pose.

In this work, we introduce a two stream hourglass network for 2D pose and a novel network for 3D hand pose regression using graph convolutional networks. To train deep learning architectures, datasets are highly significant and there exist quite a few benchmarks for hand object pose estimation. Most of the benchmarks rely on a manual annotation process and it is quite tedious, time-consuming, and costly. We also introduce a new semi-automatic labeling process for 3D hand pose estimation.

[a] https://orcid.org/0000-0001-7339-8425

Bandi, C., Kisner, H. and Thomas, U.
3D Hand and Object Pose Estimation for Real-time Human-robot Interaction.
DOI: 10.5220/0010902400003124
In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 4: VISAPP*, pages 770-780
ISBN: 978-989-758-555-5; ISSN: 2184-4321

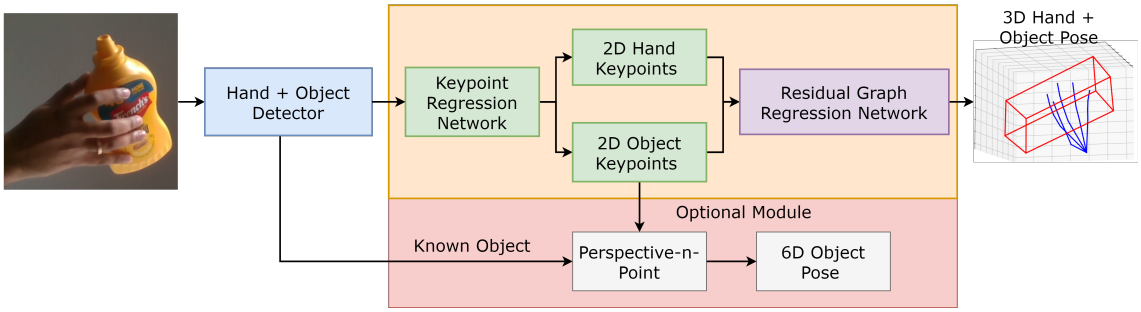

Figure 1: The overview of complete pipeline. The input is an RGB image and output is the 2D hand pose, 2D object corners, 3D object corners, 3D hand pose and an optional module to get 6D object pose using perspective-n-point algorithm.

2 RELATED WORK

In this section, we review the deep learning-based state-of-the-art techniques for 3D hand pose estimation, 6D object pose estimation, unified reconstruction of a hand-object pose, and well-known benchmarks.

2.1 3D Hand Pose Estimation and 6D Object Pose Estimation

The earliest work to estimate 3D hand pose using deep learning (Zimmermann and Brox, 2017) propose a cascaded architecture consisting of segmentation network, pose network, and pose-prior network. Segmentation network localizes the region of the hand, the segmented region of the interest is forwarded to pose network to obtain 2D score maps of hand joints and finally 2D keypoints are lifted to 3D using a pose-prior network with viewpoints. The architecture is evaluated on the rendered hand pose dataset which is purely synthetic and suffers from generalization issues on real images. To obtain better generalization, the work (Mueller et al., 2018) proposes a technique that transforms the synthetic dataset such that it resembles real-world images. Later, the images are forwarded to the regression network to regress both 2D heatmaps and 3D joint coordinates. For better generalization, 3D hand kinematics are combined with convolutional architecture. The work (Iqbal et al., 2018) introduces an approach that leverages depth information in addition to RGB image. In the first stage, an encoded-decoder network produces 2D heatmaps and latent depth maps and in the next stage, 2D pose information from heatmap is combined with depth maps to obtain normalized 3D coordinates and finally reconstructed to retain 3D pose. The direct 2D or 3D keypoints cannot express the shape of the hand so the work in (Ge et al., 2019) proposes graph convolutional neural network-based (Graph-CNN) architecture to recover the 3D mesh of the hand beside the 2D and 3D hand pose. The architecture is tested on both real and synthetic datasets. Based on the idea of using 2D hand pose to obtain 3D hand pose is further exploited in these works (Bandi and Thomas, 2020; Zhang et al., 2020).

6D Object Pose Estimation. Most of the state-of-the-art works follow two-staged processes for 6D object pose estimation. In the first stage, a CNN architecture is utilized to detect the 2D keypoints of 3D projected corners and then the perspective-n-point (PnP) (Lepetit et al., 2009) solver to compute 6D pose features. For the first stage, (Liu et al., 2016) proposes an architecture inspired by the YOLO model for 2D keypoint detection and (Peng et al., 2018) presents a pixel-wise voting scheme to further enhance the accuracy of 2D keypoint detection like RANSAC. It is possible that the detected keypoints are not completely accurate due to occlusions, which is further refined by deep iterative matching (Li et al., 2018). The 6D object poses for semantic grasping (Tremblay et al., 2018) works in a similar two-staged process of 2D keypoint detection using a CNN architecture and PnP-based object pose estimation. The model is completely trained on a synthetic dataset, generalized well on real-world images and the model is quite simple and applicable in real-time. Further improvement is shown in (Wang et al., 2019) by fusing both RGB and depth information. Pixel-wise 3D coordinate prediction of objects without textured models is presented in (Park et al., 2019) followed by PnP algorithm with RANSAC. The Multiview multi-object pose estimation (Labbé et al., 2020) is very robust and it is handled in three stages. In the first stage, 2D regions and respective initial 6D object poses are obtained for all views and then these objects are matched to recover a single consistent view and objects in the scene and camera poses are refined globally.

Unified Hand + Object Pose Estimation. The

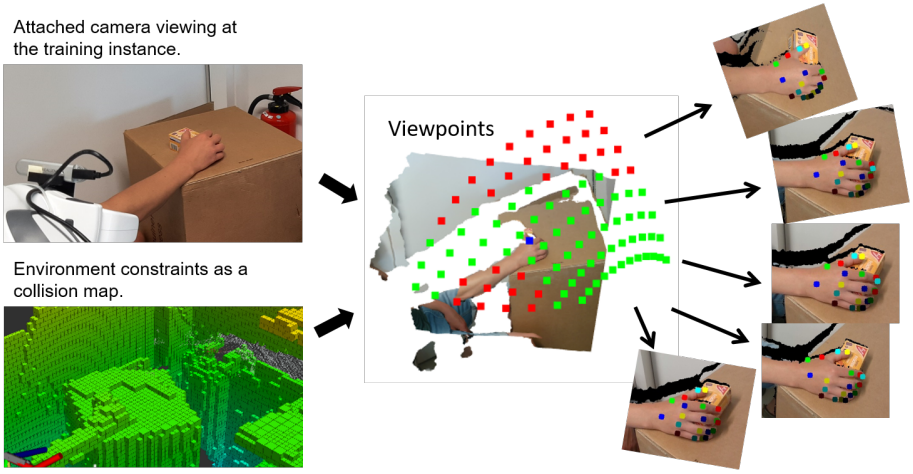

Figure 2: The pipeline of the training dataset generator. A camera is attached to the tool center point (TCP) of a robot arm and captures the scene. The current environment is included as a collision map. New camera viewpoints can be calculated with the captured information. There are reachable viewpoints (green) and unreachable viewpoints (red) as the robot motion is restricted. The selected center point (blue) is used for viewpoint calculation. From each reachable viewpoint, point clouds, as well as images, can be captured and used for the semi-automatic labeling process.

single-shot neural network (Tekin et al., 2019) recognizes the 3D hand pose and 3D object pose. In addition to 3D pose features, the network recognizes the action performed. This network drops the idea of computing 2D and 3D correspondences to compute 6D pose using PnP and instead results from the direct 3D coordinates of the bounding box around the 3D object. For applications like the handover of objects or grasp applications, the 3D keypoint representation might be insufficient. For such application, hand and object meshes are reconstructed (Hasson et al., 2019). The drawback is that the architecture trained on purely synthetic data and the retrieved object meshes are not refined. To further improve the accuracy Graph UNet (Doosti et al., 2020) based architecture is proposed. The initial 2D keypoints detections are refined using graph convolutional networks and then adaptive UNet transforms them to 3D predictions of both hand and 3D object pose.

2.2 Hand + Object Benchmarks

There exist quite a few datasets for 3D hand pose and 6D object pose estimation. 3D Hand pose is annotated on real images using manual (Sridhar et al., 2016; Mueller et al., 2017), semi-automatic (Zimmermann et al., 2019), complete automatic (Simon et al., 2017) annotation processes, and on synthetic (Mueller et al., 2018; Zimmermann and Brox, 2017) images using automated process. 6D object benchmarks like LINEMOD (Hinterstoisser et al., 2012) dataset is manually labeled, and it consists of around 1100 frames containing 15 texture-less objects. Later a large-scale YCB video dataset (Xiang et al., 2018) with a semi-automated annotation process for 6D object pose estimation is introduced with 21 textured objects containing 80 video sequences for training and 12 video test videos. This dataset is widely applied in many state-of-the-art works.

For hand object manipulations, several datasets with hand and object pose have been proposed. (Hasson et al., 2019) introduce an ObMan dataset with hands grasping objects. The large-scale dataset consists of 150,000 images that are synthetically generated. First-person hand action (Garcia-Hernando et al., 2018) dataset provides hand object interaction of daily actions with 3D hand joints and object pose. The drawback is that the dataset is captured by attaching magnetic sensors to the subject's hand and object, which in turn modifies the appearance features on RGB images. Very recently HO-3D (Hampali et al., 2019) dataset is open-sourced with hand object interactions using the objects from the YCB dataset. The dataset consists of highly occluded automatic annotations, and it is very suitable for real-world hand object interaction scenarios. The ContactPose (Brahmbhatt et al., 2020) dataset is also a large-scale dataset containing 2.9 million images of human hands grasping objects with contact maps and this dataset is captured using 3 Kinectv2 RGB-D cameras with known household objects with markers. In (Ye et al., 2021), object handovers from human to human is extensively evaluated and also introduces a dataset with 18k handover videos.

3 SEMI AUTOMATIC TRAINING INSTANCES

The hand pose estimation depends on the quality of training data from the environment, e.g. a higher data variety increases robustness. If data with later used camera sensors exist, the performance increases in a real-world scenario. However, the creation of labeled training data is tedious and time-consuming so we automatize parts of the process to decrease creation time. For this purpose, we need sensors to detect camera poses, as well as a concept for camera movements. This leads to the idea of using a robot arm and attaching a camera to it. The following section describes the setup and the data creation pipeline in more detail.

3.1 Scenario Setup

We need to change and measure camera poses, as flexible and accurate as possible, to ensure a high training data quality. Thus, our pipeline is based on a robot arm and a RGB-D camera sensor. The robot shall move the camera to different poses to get different viewing angles while measuring the position of the camera. The TCP of the robot includes maximum freedom of movement and thus we attach a camera near to it. (The exact camera pose is determined by robot sensors and a camera calibration which has to be executed beforehand. Furthermore, the robot has to know its environment to guarantee collision freeness. Thus, our robot controller includes a static map of the environment. Furthermore, the current camera view is added to the collision map. However, more details on collision maps and robot control are out of the scope of this work.

3.2 Pipeline

Our semi automatized dataset creation pipeline consists of two main stages. The first stage is used to capture images and generate new camera poses. The saved data is forwarded to the second stage, a post-processing pipeline that uses all captured information to generate training data. Figure 2 visualizes the pipeline with a sample scenario.

3.2.1 Move and Capture

At first, we put the object of interest near the robot and align the camera to the object [1]. We want to generate landmarks for hand-tracking, thus our object of interest is a human hand. After the first captured image and pose, the hand has to stay as static as possible [2]. Now, a computer transforms the captured information into a visualizer using the robot frame as the origin. An external operator uses a GUI to set a point as the center point for the next camera viewpoints. Next, the object with respect to the center point has to be captured from different viewing angles. However, we could use random camera poses. Instead, we need a predictable motion of the robot to increase the safety level. We define a sphere with the initial camera pose. The resulting sphere corresponds to a set of infinite viewpoints around the center point. We discretize the viewpoints using a pre-defined angle distance, e.g. 10°. The resulting points are the next camera poses in world coordinates. The pipeline captures for each of the viewpoints the coordinates of the camera as well as a 3D point cloud. The camera pose is used to transform the information into the robot coordinate system; respectively all point clouds are transformed into the same coordinate system leading to a stitched point cloud. Next, we post-process the captured data.

3.2.2 Post Processing

The post-processing uses the point clouds, captured 2D images as well as camera transformations as input data. Additionally, a template of desired landmarks is required. It is not guaranteed that all landmarks are visible in each image, e.g. restricted camera view angle, occluded by objects. We overcome hidden landmarks by displaying the complete aligned point cloud in a GUI as we need exact training data. Thus, an external expert has to pick in the GUI initial landmarks, e.g. the fingertips, in the aligned cloud concerning the pre-defined landmark template. The selection results in a set of point indices, that are set as ground truth information. From that, the algorithm iterates through the captured point clouds and camera poses. We search for each ground-truth the corresponding point in the specific point cloud and set the extracted indices as the landmark. However, if the euclidean distance between the ground truth point and the new landmark is larger than the threshold $\tau = 0.05cm$, we define the landmark as 0 corresponding to an occluded or invisible point. The algorithm outputs a 3D point cloud, a corresponding 2D image, the indices as well as the positions of the landmarks in 2D and 3D. We name our dataset as robot arm semi-automatic hand dataset (RASH). Adding ground truth data for bounding boxes differs from the normal land-

[1] We assume the object and the robot to be placed in free space, to guarantee that the robot can move around it (considering the movability limitations of the robot).

[2] Proposing an external fixation device or something similar is out of scope of this work.

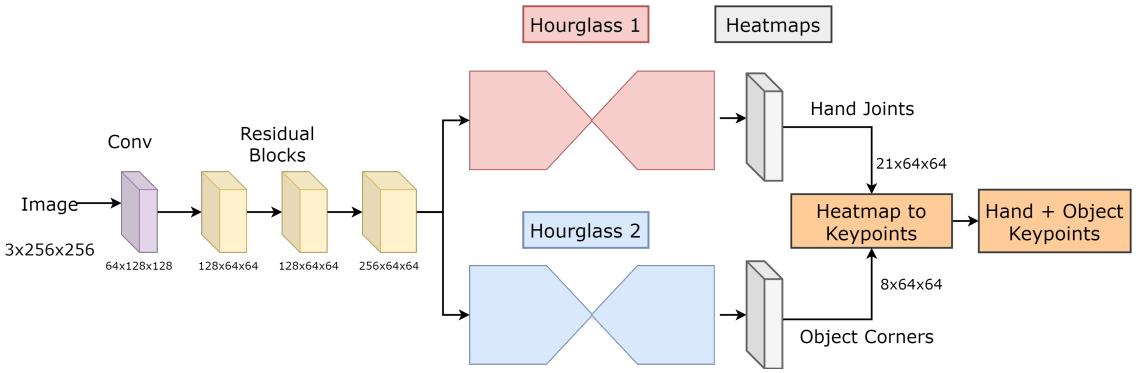

Figure 3: The architecture of 2D hand pose and 2D object corner regression using heatmaps.

marks as selecting bounding boxes in point clouds is not intuitive. We use 2D images of the initial cloud for selecting 2D bounding boxes. However, we need to add depth information, to convert 2D box information into 3D. Therefore, the depth of the object has to be predefined, e.g. a hand has a size of about 20cm. The 2D box relates to a set of indices corresponding to 3D points. We select the nearest point from the extracted points and add the pre-defined object size. Thus, the external operator selects x and y coordinates of the bounding box edges, and the depth is automatically added. The bounding box edges can be similarly used as ground truth landmarks. Using this setup, we capture 12000 hand instances which allows maximum error of 0.5mm for training the network.

4 METHODOLOGY

In this section, we introduce the architecture for 3D hand pose regression and object pose estimation. Given an RGB image $I = R^{height \times width \times 3}$, we regress 2D Hand pose $P_{2D} = R^{21\times 2}$, 2D object keypoints $O_{2D} = R^{8\times 2}$, 3D hand pose of size $P_{2D} = R^{21\times 3}$ and 3D object keypoints $O_{3D} = R^{8\times 3}$.

4.1 Hand + Object Detection

Initially, we consider an independent object detection model to detect hands and known objects that are used in the interaction environment. The object detection architectures are widely researched for many real-time applications and there exist one staged (Bochkovskiy et al., 2020; Liu et al., 2016) and two-staged models (Dai et al., 2016) with distinct backbones either for GPU (He et al., 2015) or for CPU platforms (Sandler et al., 2018). In this work, we reuse the well-known YOLOv4 (Bochkovskiy et al., 2020) architecture with MobileNet (Sandler et al., 2018) backbone for training the region of hands and objects. As the network is quite fast and it does not affect the speed of the cascaded network.

4.2 2D Hand Pose and Object Keypoints

Once the region of hand and object is detected, we pass the cropped region to heatmap regression network (HRN) to estimate the 21 hand joints and 8 object corners. To regress keypoints using heatmaps, an encoder-decoder or hourglass based architecture is considered. The hourglass based model with residuals introduced in (Newell et al., 2016) works well for human pose estimation and later adopted to hand based keypoint regression models. Based on that, we introduce a two-stream hourglass model to regress heatmaps of hand and object joints. The HRN is clearly depicted in Figure 3. As the RGB image contains both hand and object, we extract common features using convolutions and residuals (He et al., 2015). Later, the information is shared to hourglass 1 to extract hand joint heatmaps and hourglass 2 to extract object corners. The network features before hourglass are $3 \times 256 \times 256 \rightarrow conv(64 \times 128 \times 128) \rightarrow residual(128 \times 64 \times 64) \rightarrow residual(128 \times 64 \times 64) \rightarrow residual(256 \times 64 \times 64))$. The features are directly passed through two hourglass models without further processing to extract heatmaps of hand joints and object corners. From the network, we obtain 21 heatmaps for each hand and 8 heatmaps for each object present in the image. Each heatmap consists of one joint with size of 64×64. The location of keypoint is the location of maximum value exists in the heatmap and the maximum location is converted to orginal image size by multipying it with 4 (i.e., $64 \times 4 = 256$). An example of heatmaps is shown in Figure 5.

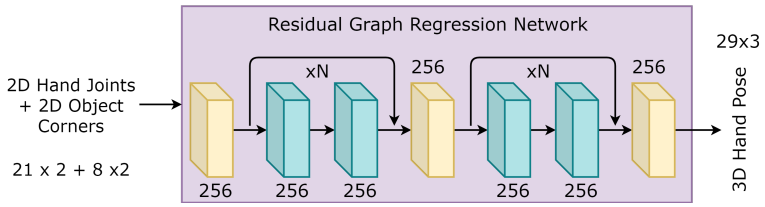

Figure 4: The architecture of residual graph regression network for 3D pose regression.

4.3 Residual Graph Regression Network (RGRN)

We will revisit briefly the concept of GCNs as proposed in (Kipf and Welling, 2016). The aim is to learn features on a graph structure data and convolutional filters are shared across all graph locations. The graph structure is represented in the form of an adjacency matrix $A \in K \times K$, where K is the number of input nodes. The input features of every node are represented in the form of a feature matrix $K \times H$, where H is the input features. The output features $f(H,A)$ of each layer l are obtained by multiplying the adjacency matrix with input nodes and trainable weights W, and features are then passed to non-linear functions like ReLU(Agarap, 2019).

$$f(H^{(l)},A) = \sigma(AH^{(l)}W^{(l)}) \quad (1)$$

As our hand model is in the form of a graph, we use basic building blocks of graph convolutions and combine them with residual connections (He et al., 2015). To learn the graph structure, we use the predefined kinematic structure of hand and object joints as an adjacency matrix to the graph network. Given 2D keypoints of shape, 29×2 as input, each graph convolution output 256 features and the final output layer regress 29×3 features i.e., the 3D coordinates of the hand pose and object pose. The output of each graph convolution is normalized and passed through ReLU (Agarap, 2019) non-linear function. As the input features are wrist-centric (i.e., the wrist joint is (0,0,0)), we also introduce bias during training. The purple block in Figure 4 represents the RGRN model for 3D hand pose regression. As the number of residual layers N is not fixed, we experiment with the value of N during the training process for best performance. Each block consists of graph convolution operation and the network layers are as follows $21 \times 2 \rightarrow 21 \times 256 \rightarrow N - Residual(21 \times 256 \rightarrow 21 \times 256) \rightarrow 21 \times 256 \rightarrow N - Residual(21 \times 256 \rightarrow 21 \times 256) \rightarrow 21 \times 3$. See Section 7 for training and validation stability.

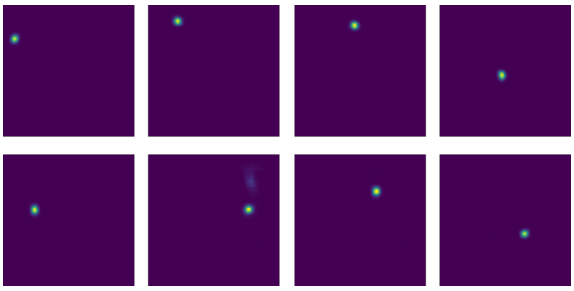

Figure 5: An example of 64×64 heatmap of each joint.

4.4 Network Loss

To train a model, we must compute loss for backpropagation, the heatmap loss for 2D hand pose l_{hand2D}, the 2D object corner l_{obj2D}, and the pose loss for 3D hand pose l_{hand3D} and 3D object corner pose l_{obj3D}. For both 2D and 3D loss computation, mean squared error (MSE) loss is utilized. The individual loss computation is

$$l_n = ||groundtruth - predicted||^2 \quad (2)$$

Where n is l_{hand2D}, l_{obj2D}, l_{hand3D}, and l_{obj3D}.

The overall heatmap loss $L_{heatmap}$ for 2D heatmap regression network is computed as

$$L_{heatmap} = l_{hand2D} + l_{obj2D} \quad (3)$$

The overall 3D pose loss $L_{hand3D+Object3D}$ is computed as

$$L_{hand3D+Object3D} = l_{hand3D} + l_{obj3D} \quad (4)$$

5 EXPERIMENTS

5.1 Datasets

In this section, we extensively experiment with model hyperparameters and test the generalizability of the proposed model on two open source datasets in which HO3D dataset (Hampali et al., 2019) contains both hand manipulating objects, FreiHand dataset (Zimmermann et al., 2019) contains hand pose with self-occlusions, and RASH dataset: 1) HO-3D dataset (Hampali et al., 2019): as the dataset contains both hand and object annotations. HO-3D dataset contains

Table 1: Training Parameters.

Parameter	Object Detector	2D Hand and Object Pose	3D Hand and Object Pose
Network	YOLOv4	Hourglass	RGRN
Input size	416×416	256×256	29×2
Output	Bounding Box	$64 \times 64 \times 21 + 8 \times 2$	29×3
Epochs	120	200	200
Learning rate	0.0001	0.0001	0.0001/0.001
Optimizer	Adam	RMSprop	RMSprop/Adam(Kingma and Ba, 2017)
Framework	PyTorch		
GPU	Nvidia RTX 2060 Super		

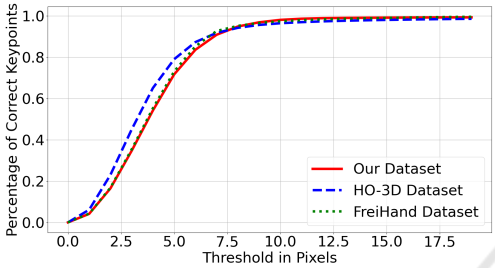

Figure 6: The percentage of correct keypoints of 2D hand pose on three different datasets.

hands manipulating objects. Although the dataset consists of 77,558 samples only 66,034 samples are available with full hand annotations. So, we randomly split such that 50,000 samples are used for training, 16,325 samples for validation purposes and the remaining 11,524 unlabeled data for evaluation. The process of hand-object labelling (Hampali et al., 2019) is completely automated with less initialization constraints. 2) FreiHand dataset (Zimmermann et al., 2019) is a semi-automatic annotated hand dataset consisting of over 32,000 training samples and 4,000 test samples. Although the dataset is highly occluded, no object pose annotations are present. 3) RASH dataset is a small-scale hand dataset captured using Panda robot arm consisting of 10,000 training instances and 2,000 test samples. The images are captured using Intel Realsense 415 camera with a resolution of 640×480 pixels.

5.2 Evaluation Metrics, Training Parameters, and Results

We evaluate the performance of 2D pose, and 3D pose using three different metrics:

1) The mean distance error between the predicted keypoints and the groundtruth. 2) Percentage of correct keypoints (PCK) is an exceedingly popular evaluation metric for both 2D and 3D pose. PCK considers a keypoint as correct if it falls under a certain threshold, with pixel value threshold for 2D and millimeter distance for 3D around the ground truth. 3) The area under the curve (AUC) of the PCK graph computed for a threshold of 50mm.

Table 2: Evaluation of euclidean position error (EPE) in millimeters on validation set.

Dataset	Residual blocks	
	N=2	N=3
HO-3D (Hampali et al., 2019)	13.4	9.2
FreiHand (Zimmermann et al., 2019)	13.2	8.8
Ours (RAH)	10.1	8.2

The complete training parameters are listed in Table 1 and the models are trained independently depending on the dataset. We evaluate the proposed model by experimenting with hyperparameters such that the best possible accuracy is acheived. In addition to training hyperparameters, we need to consider the architecture parameters like number of stacks and hidden layers.

5.2.1 2D Hand and Object Pose

For 2D hand pose estimation, we consider PCK evaluation metric and it is plotted in Figure 6. On HO-3D (Hampali et al., 2019) validation dataset the value of PCK @ 10 pixel is ≈ 0.969 and on RASH dataset PCK@10 pixel is ≈ 0.986. The difference is due to the high occlusion in HO-3D (Hampali et al., 2019)

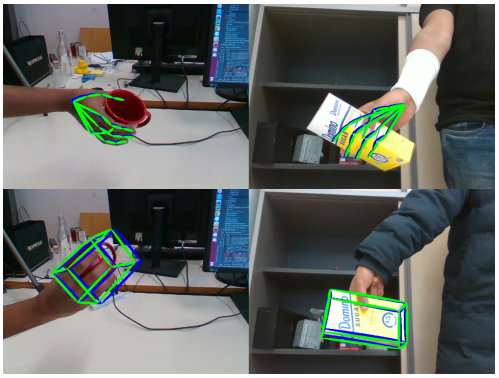

Figure 7: The 2D hand pose and object pose outputs.The green lines represent groundtruth and the blue lines indicate predicted keypoints.

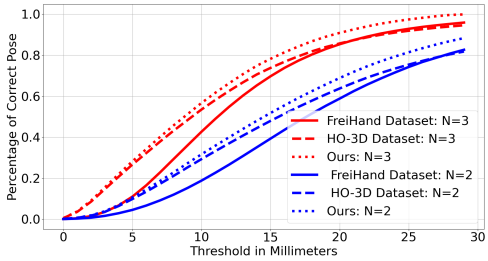

Figure 8: Percentage of correct keypoints (PCK) evaluation of 3D Hand pose.

Table 3: Comparison to the state-of-the-art.

Method	Area Under Curve (AUC) %
HO-3D (Hampali et al., 2019)	79.0
FreiHand (Zimmermann et al., 2019)	79.1
Ours on HO-3D	79.4
Ours on FreiHand	84.2

dataset compared to RASH dataset. Similarly, 2D PCK of object keypoints on HO-3D dataset is ≈ 0.9 for 10 pixel threshold. The 2D hand and object pose on HO3D validation samples can be observed in Figure 7.

5.2.2 3D Hand and Object Pose

We train the 2D pose and 3D pose networks independently without sharing the weights. RGRN (see Figure 4) has hyperparameter N that determines number of hidden residual layers. The network is trained by setting the value of N to 2 and 3. The 3D PCK graph is represented in Figure 8. From the graph, we can clearly observe that by increasing the number of hidden layers performance is improved. In addition to PCK metric, we evaluate mean euclidean 3D position error (EPE) by setting $N = 2$ and $N = 3$. The EPE for HO-3D (Hampali et al., 2019) dataset, FreiHand dataset (Zimmermann et al., 2019), and RASH dataset is represented in Table 2. Few results on validation dataset are mentioned in Figure 9. Finally, we test the accuracy of our proposed model on HO-3D (Hampali et al., 2019) and FreiHand dataset (Zimmermann et al., 2019) datasets and compare it to the state-of-the art. For testing the accuracy on unlabeled data, we have to follow the instructions by (Hampali et al., 2019) to submit the outputs to a submission board. The Table 3 represents the AUC in percentage computation of 3D PCK curve with the threshold of 50mm and we can clearly see that the architecure performs well. Although the architecture performs better on both the datasets, the AUC computation of ContactPose(Brahmbhatt et al., 2020) dataset has shown to be even higher and the difference is due to the fact that the hand mesh (3D MANO) (Romero et al., 2017) models are used to estimate the keypoint errors and in this work we use direct 3D regression.

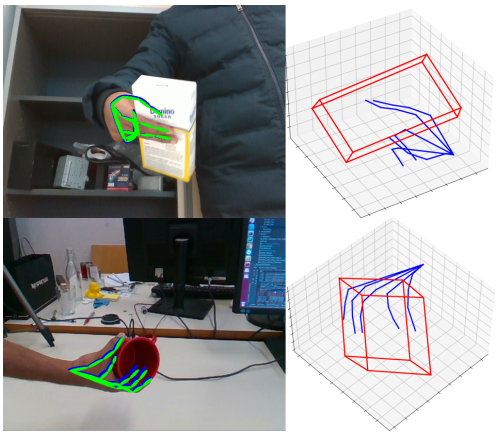

Figure 9: 3D hand and object pose on HO3D validation samples.

5.3 Discussion and RASH Dataset

In robotics, there exist applications like handover of objects from humans to robots and viceversa. In this work, our aim is to build an architecture that is suitable for bidirectional handover applications. As the image resolution and occlusions are higher in HO-3D dataset, we first trained proposed model completely on HO-3D dataset and evaluated in human-robot interaction environment as in Figure 10. The resulting 3D hand pose and 2D hand pose in human-robot interaction environment can be oberved in Figure 11. By just using HO-3D dataset in our environment, we noticed that the 3D hand pose have high errors when there is no object in the hand (see Figure 11, left image with 3D and 2D). Figure 11 right image represents hand with object in the human robot interaction environment. Although HO-3D dataset is a large-scale and highly occluded dataset, it does not contain enough instances for hands without objects. To make our architecture work well for bidirectional handovers, we collected our own dataset known as RASH

Figure 10: Human-robot interaction environment for handover of objects.

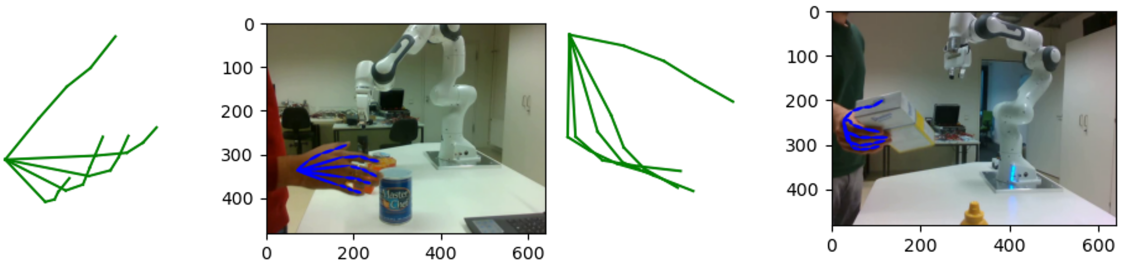

Figure 11: Test ouputs in human-robot interaction environment; model trained on HO3D dataset.

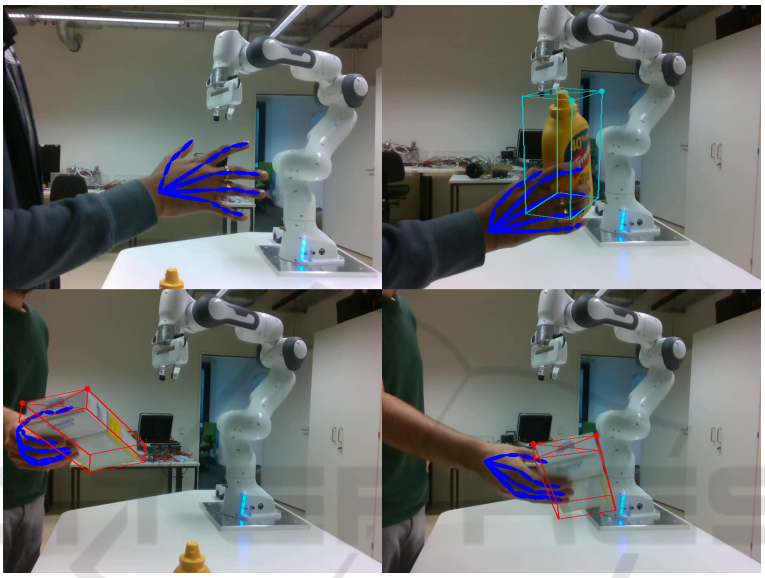

Figure 12: Hand and object pose estimation in human-robot interaction environment.

dataset. The dataset consists of mostly open hand images with very low occlusions. To enchance the performance, we combined training instances of HO-3D dataset with RASH dataset and retrained the proposed model. The Figure 12 represents the few pose samples of hands without objects and hands manipulating objects in human-robot interaction environment. From that we noticed a significant improvement for 3D hand poses (hands without objects). The number of parameters in the complete architecture adds upto 20.26 million. During the processs of training, we utilize Nvidia 1080 Ti graphical processor with 12 gigabyte memory. For inference, we use Nvidia 1660 Ti processor with 6 gigabyte memory. The complete pipeline acheived a framerate of $\approx$ 16 fps on single GPU without visualization. It is further possible to improve the framerate by reducing the stacks in hourglass but it compromises the accuracy of 2D keypoint estimation. For real-time applications, the training dataset must be highly occluded. To test this proposed pipeline in real-world application, we retrained the model by combining HO-3D dataset and RASH dataset. To avoid the tedious process of manual labelling, we opt for a semi automatic labelling and we are also currently working on fully automated annotation process using robot arm without any constraints.

6 CONCLUSIONS

We proposed a complete pipeline to regress 2D pose, 3D hand pose and object pose. The 2D hand pose is estimated using hourglass architecture and the intermediate features from 2D hourglass network is further extended to regress the object keypoints in the image. To regress 3D hand pose from 2D hand pose, we introduced a residual graph regression network with N residual connections and achieved best performance for $N = 3$. We evaluated the proposed network with three different metrics on three different datasets. We will further focus on improving the object keypoint regression, accuracy of training instances and forward the obtained information to a grasp planner for real-time handover of object interactions.

ACKNOWLEDGEMENTS

This work is Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project-ID 416228727 - SFB 1410.

REFERENCES

Agarap, A. F. (2019). Deep learning using rectified linear units (relu).

Bandi, C. and Thomas, U. (2020). Regression-based 3d hand pose estimation using heatmaps. In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,*, pages 636–643. INSTICC, SciTePress.

Bochkovskiy, A., Wang, C., and Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934.

Brahmbhatt, S., Tang, C., Twigg, C. D., Kemp, C. C., and Hays, J. (2020). ContactPose: A dataset of grasps with object contact and hand pose. In *The European Conference on Computer Vision (ECCV)*.

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks.

Doosti, B., Naha, S., Mirbagheri, M., and Crandall, D. J. (2020). Hope-net: A graph-based model for hand-object pose estimation. *CoRR*, abs/2004.00060.

Garcia-Hernando, G., Yuan, S., Baek, S., and Kim, T.-K. (2018). First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.

Ge, L., Ren, Z., Li, Y., Xue, Z., Wang, Y., Cai, J., and Yuan, J. (2019). 3d hand shape and pose estimation from a single RGB image. *CoRR*, abs/1903.00812.

Hampali, S., Oberweger, M., Rad, M., and Lepetit, V. (2019). HO-3D: A multi-user, multi-object dataset for joint 3d hand-object pose estimation. *CoRR*, abs/1907.01481.

Hasson, Y., Tekin, B., Bogo, F., Laptev, I., Pollefeys, M., and Schmid, C. (2020). Leveraging photometric consistency over time for sparsely supervised hand-object reconstruction. *CoRR*, abs/2004.13449.

Hasson, Y., Varol, G., Tzionas, D., Kalevatykh, I., Black, M. J., Laptev, I., and Schmid, C. (2019). Learning joint reconstruction of hands and manipulated objects. *CoRR*, abs/1904.05767.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. (2012). Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part I*, ACCV'12, page 548–562, Berlin, Heidelberg. Springer-Verlag.

Iqbal, U., Molchanov, P., Breuel, T., Gall, J., and Kautz, J. (2018). Hand pose estimation via latent 2.5d heatmap regression. *CoRR*, abs/1804.09534.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Labbé, Y., Carpentier, J., Aubry, M., and Sivic, J. (2020). Cosypose: Consistent multi-view multi-object 6d pose estimation. *CoRR*, abs/2008.08465.

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp: An accurate o(n) solution to the pnp problem. *Int. J. Comput. Vision*, 81(2):155–166.

Li, Y., Wang, G., Ji, X., Xiang, Y., and Fox, D. (2018). Deepim: Deep iterative matching for 6d pose estimation. *CoRR*, abs/1804.00175.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37.

Mueller, F., Bernard, F., Sotnychenko, O., Mehta, D., Sridhar, S., Casas, D., and Theobalt, C. (2018). Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.

Mueller, F., Mehta, D., Sotnychenko, O., Sridhar, S., Casas, D., and Theobalt, C. (2017). Real-time hand tracking under occlusion from an egocentric rgb-d sensor. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937.

Park, K., Patten, T., and Vincze, M. (2019). Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. *CoRR*, abs/1908.07433.

Peng, S., Liu, Y., Huang, Q., Bao, H., and Zhou, X. (2018). Pvnet: Pixel-wise voting network for 6dof pose estimation. *CoRR*, abs/1812.11788.

Romero, J., Tzionas, D., and Black, M. J. (2017). Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):245:1–245:17.

Rosenberger, P., Cosgun, A., Newbury, R., Kwan, J., Ortenzi, V., Corke, P., and Grafinger, M. (2020). Object-independent human-to-robot handovers using real time robotic vision. *CoRR*, abs/2006.01797.

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. (2018). Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381.

Simon, T., Joo, H., Matthews, I. A., and Sheikh, Y. (2017). Hand keypoint detection in single images using multiview bootstrapping. *CoRR*, abs/1704.07809.

Sridhar, S., Mueller, F., Zollhoefer, M., Casas, D., Oulasvirta, A., and Theobalt, C. (2016). Real-time joint tracking of a hand manipulating an object from rgb-d input. In *Proceedings of European Conference on Computer Vision (ECCV)*.

Tekin, B., Bogo, F., and Pollefeys, M. (2019). H+O: unified egocentric recognition of 3d hand-object poses and interactions. *CoRR*, abs/1904.05349.

Tekin, B., Sinha, S. N., and Fua, P. (2017). Real-time seamless single shot 6d object pose prediction. *CoRR*, abs/1711.08848.

Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., and Birchfield, S. (2018). Deep object pose estimation for semantic robotic grasping of household objects. *CoRR*, abs/1809.10790.

Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., and Savarese, S. (2019). Densefusion: 6d object pose estimation by iterative dense fusion. *CoRR*, abs/1901.04780.

Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2018). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes.

Yang, W., Paxton, C., Mousavian, A., Chao, Y., Cakmak, M., and Fox, D. (2020). Reactive human-to-robot handovers of arbitrary objects. *CoRR*, abs/2011.08961.

Ye, R., Xu, W., Xue, Z., Tang, T., Wang, Y., and Lu, C. (2021). H2O: A benchmark for visual human-human object handover analysis. *CoRR*, abs/2104.11466.

Zhang, Y., Chen, L., Liu, Y., Zheng, W., and Yong, J. (2020). Explicit knowledge distillation for 3d hand pose estimation from monocular rgb. In *BMVC*.

Zimmermann, C. and Brox, T. (2017). Learning to estimate 3d hand pose from single rgb images. In *IEEE International Conference on Computer Vision (ICCV)*. https://arxiv.org/abs/1705.01389.

Zimmermann, C., Ceylan, D., Yang, J., Russell, B. C., Argus, M., and Brox, T. (2019). Freihand: A dataset for markerless capture of hand pose and shape from single RGB images. *CoRR*, abs/1909.04349.

APPENDIX

Pipeline for Handover of Objects. In human-robot interaction environment, it is possible that only hand is present in the scene. As the network designed in section 4 is for both hand and object images, we included another stream if only hand is present in the scene. The overall pipeline can be observed in Figure 13.

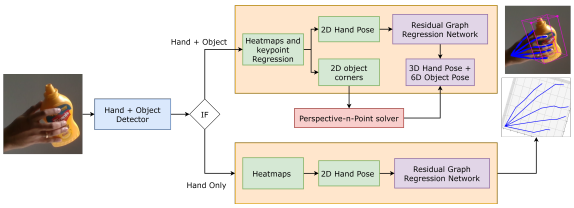

Figure 13: The architecture for bidirectional handovers.

Training and Validation Loss. The training and validation loss of the proposed RGRN architecture can be seen in Figure 14. From the Figure, we can clearly notice the stability of both training and validation loss.

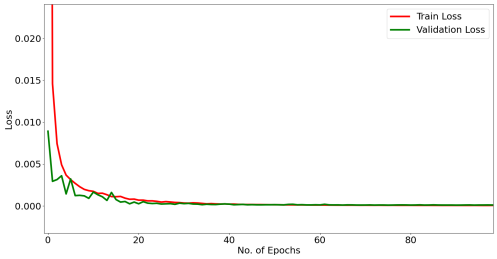

Figure 14: Training and validation loss of RGRN architecture.

Detection Output. The output of YOLOv4 (Bochkovskiy et al., 2020) on hands and objects can be observed in Figure 15.

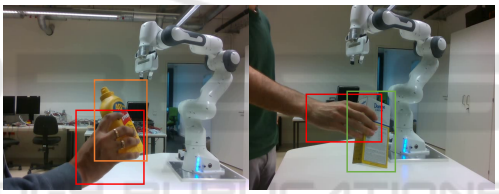

Figure 15: The outputs of YOLOv4 architecture for hand and object detection.