Semi-automatic 3D Object Keypoint Annotation and Detection for the Masses

Kenneth Blomqvist*, Jen Jen Chung*, Lionel Ott* and Roland Siegwart*
* Autonomous Systems Lab, ETH Zürich

Abstract—**Creating computer vision datasets requires careful planning and lots of time and effort. In robotics research, we often have to use standardized objects, such as the YCB object set, for tasks such as object tracking, pose estimation, grasping and manipulation, as there are datasets and pre-learned methods available for these objects. This limits the impact of our research since learning-based computer vision methods can only be used in scenarios that are supported by existing datasets. In this work, we present a full object keypoint tracking toolkit, encompassing the entire process from data collection, labeling, model learning and evaluation. We present a semi-automatic way of collecting and labeling datasets using a wrist mounted camera on a standard robotic arm. Using our toolkit and method, we are able to obtain a working 3D object keypoint detector and go through the whole process of data collection, annotation and learning in just a couple hours of active time.**

I. INTRODUCTION

Most modern computer vision methods use large datasets to learn to predict features at run time. These have been demonstrated to enable many new capabilities in robotic object manipulation. While the methods are impressive, they are data hungry and require sizeable datasets of ground truth annotations to train. If we could quickly and cheaply create datasets, we could expand to more environments and enable many downstream tasks.

The data requirements force researchers of downstream robotics tasks to either use standard objects, for which trained models and computer vision pipelines have been made available, or a large investment has to be made upfront to collect and label a dataset. Creating a dataset requires either hand-labeling thousands of frames one-by-one, having a data collection setup with environment markers, as done in [1], or a tool such as LabelFusion [2] can be used to partially automate the annotation process. However, LabelFusion requires mesh models of the objects. Creating a known model for objects in turn requires a high-fidelity object scanning setup, which is often unavailable. It also requires the objects to be rigid, or additional parameters need to be estimated to model deformation. Additionally, the objects and environment have to be such that depth sensors are able to accurately measure depth, excluding reflective or transparent objects.

In this paper, our goal is to track category-level semantic points in an object's coordinate frame relative to the camera frame for downstream robotic manipulation tasks. "Category-level" meaning that objects vary, but the intra-category semantic meaning of keypoints are the same. Specifically, we want a system with the following properties:

1) Can estimate 3D object keypoints on arbitrary objects
2) Requires little effort to handle novel objects
3) Can be used in the wild without having to use markers, motion tracking systems or otherwise modify the environment
4) Does not rely on accurate depth sensing
5) Can track multiple objects simultaneously in the image frame

Existing methods, such as PVN3D [3] and kPAM [4] require semantic segmentation maps to train or they rely on external object instance segmentation. Semantic segmentation maps are time consuming to annotate, making the systems more expensive to deploy in new scenarios and for new objects.

In contrast, we present a complete 3D object keypoint tracking system, including both a learning-based object keypoint algorithm and a method to very quickly obtain the training labels needed by the algorithm. Our method builds on the insight that we can forgo using semantic segmentation maps to distinguish between objects, if we instead introduce a center keypoint and predict a center map that associates each keypoint with a center keypoint. The amount of objects in the scene is inferred from the amount of detected center keypoints. This makes the labeling task a lot faster, as we can simply label 2D keypoints instead of having to also create dense instance segmentation masks.

We present a way to speed up data collection by capturing many views of the scene and propagating labels from two labeled viewpoints to all the others. We show that by calibrating our robot and making use of calibration and the kinematics of a robot arm, we can forgo using a motion tracking system or environment markers, as done by previous works. Using our system, data can be collected in the wild wherever our robot goes. This means that our tools can be deployed directly on the hardware intended for the downstream robotic task, streamlining the full problem definition and solution by avoiding additional steps. Calibrated robots are now commonly available and by using one, the data collection can be further automated and enables collecting data autonomously.

We show two different versions of our learning-based algorithm that leverages our data collection pipeline to track keypoints of multiple objects in a scene. The first one uses both views of a stereo camera. The other one is a variation of the algorithm that can work with a monocular RGB camera.

We validate our method and tracking pipeline in experiments on two different object keypoint tracking scenarios. The first one is a single object valve tracking scenario. The second is a multiple object cup tracking task, showcasing that we can handle multiple objects simultaneously in a frame. We

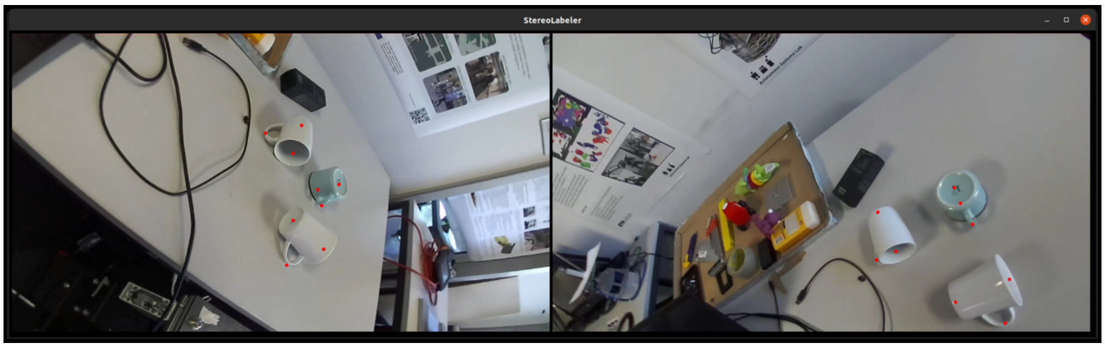


Fig. 1. StereoLabel, our keypoint labeling tool. The user is presented with two images of the scene to label. The images are selected to maximize the orthogonality of the views.

show that using only 22.5 minutes of recorded data across 45 sequences, and using less than 15 minutes of labeling time, we can learn a model that can track keypoints on objects of interest. We demonstrate that the resulting tracker is accurate enough to enable manipulation tasks, such as rotating a valve.

Code for our project is made available at `github.com/ethz-asl/object_keypoints`.

II. RELATED WORK

A. Datasets and Labeling Tools

Several object pose datasets have emerged which use ground truth meshes. The most commonly used meshes are of the YCB object set [5]. The YCB-video dataset from [6] provides labeled 6D poses for objects in RGB images. The authors demonstrated that the dataset was capable of training their PoseCNN 6D object pose estimator. An initial estimate of the object poses from PoseCNN were used to generate the YCB-M dataset [7]. This dataset was collected with seven depth cameras and they used fiducial markers and depth refinement to obtain the common frame of reference between cameras. While a robot arm was used to facilitate data collection, the authors did not use the kinematics of the robot nor did they use hand-eye calibration in the labeling process. Moreover, both of these datasets are limited to the YCB object set. [8] uses object models, simulation and rendering to obtain a dataset of ground truth object poses.

While other labeling tools exist to create datasets of a priori unknown objects, these often have other limitations. [9] provides a semi-automated tool for creating 2D and 3D bounding box labels for multi-object scenes in RGB-D video. Their algorithm uses a GrabCut-based approach [10] to interpolate annotations over timesteps. However, the user still needs to adjust the propagated bounding boxes in each of the following frames. LabelFusion [2] can handle cluttered scenes, however, object meshes are required and must either be given or created manually using a scanning routine (e.g. with a handheld scanner or turntable). Our proposed method avoids this requirement altogether.

Finally, several methods train keypoint detectors using only a small set of labeled data. Simon et al. [11] bootstrap a keypoint detection dataset for hand pose estimation using multiple views of the scene. The authors ensure that each iteration introduces new information via multiview geometry. However, because of this, performance is tied to the number of cameras in the setup. Multiview geometry is used by [12] for human and animal pose estimation by deriving a differentiable semi-supervised loss function which is equivalent to minimizing epipolar divergence. They show that they can train a keypoint detection network using a large set of unlabeled images and comparatively few labeled images.

B. Object Keypoints and 6D Pose Estimation

Methods exist which predict keypoints, in 2D or 3D, to calculate the 6D object pose. Some estimate 2D points on an RGB image and solve for the pose using a PnP algorithm [13]–[17]. Others predict keypoints directly in 3D space [3], [18]. 6-PACK [19] presents a way to track single objects in real-time using keypoints which emerge in an unsupervised way. As the keypoints are learned end-to-end, additional components such as an attention mechanism are required in their keypoint tracking pipeline. S3K [20] is a self-supervised approach to learn semantic 3D keypoints. Similar to our approach, the authors use multiple camera views to propagate labels across images. However, in their case, they require a four-camera setup while our method is designed to work with a single camera. NOCS [21] uses a representation shared within an object category. The authors learn a model to regress to this representation from RGB and depth maps. PVN3D [3] learns a model which produces semantic segmentation maps as well as per pixel keypoint and center votes from RGB-D frames. Similarly to PVN3D, we use a center prediction map to track multiple objects. However, PVN3D uses ground truth semantic instance segmentation maps to distinguish objects from each other, which are hard and expensive to label. We avoid this by associating keypoints directly with their corresponding object's center. This also circumvents the need for the expensive clustering step to aggregate pixel-wise predictions.

KPAM [4], [22] presented a way to track category-level object keypoints. However, their system can only track single objects due to the integral pose regression step it relies on [23]. Furthermore, for the same reason, it can't deal with many keypoints of the same type. KeyPose [1] is an object keypoint detection method and dataset, which also uses stereo views of a scene. This method is only applicable in single object scenes; detected keypoints are not associated to objects, which makes tracking multiple objects infeasible. Further, KeyPose only works with objects that have unique keypoints. Modeling objects such as the valve in our experiments is not possible, as it has several ambiguous keypoints. This limitation is due to the spatial softmax operation that is used in the output heatmaps. The dataset collection method proposed by KeyPose relies on fiduciary tags that are placed in the scan environment. We propose and demonstrate the feasibility of a method that does not require modifying the environment and that relies solely on a calibrated robot with a camera.

III. METHOD

Here we describe the components of our framework: the hardware setup, the procedure to collect video sequences with camera poses, our algorithm to compute ground truth labels, a stereo multiple object keypoint detection pipeline and a variation that uses only a monocular RGB camera.

A. Hardware Setup

Our method requires a calibrated image sensor along with a way to control it into different known viewpoints. For this, we use a StereoLabs ZED Mini stereo RGB camera, mounted on the wrist of a Franka Emika Panda robot arm. The robot arm has accurate encoders at each joint which give readings on the position of each joint. Using a model of the robot and the position readings, we can accurately compute the position of the wrist, relative to the base frame of our robot.

We compute the intrinsic parameters and left-to-right-camera transformation of the stereo camera using Kalibr [24]. The wrist-to-camera frame transformation we calibrate using the ethz-asl/hand_eye_calibration package [25].

B. Data Collection

For training a neural network to detect keypoints, we need a dataset of image frames and a set of keypoint locations for each image. To obtain these, we collect 30s long sequences while our robot scans the target objects, observing the objects from multiple viewpoints. We save the pose of both camera frames, relative to the base frame of the robot and the RGB frames from the left and right camera sensors. In the next section, we describe how we obtain the keypoints in image coordinates for each image.

C. StereoLabel: Labeling and Generating a Dataset

For each object category of interest, we define a set of keypoints that is most convenient for manipulating objects from the category. For example, in the case of coffee cups, we can define the keypoints to be the bottom center, center top and the outermost point on the handle of the cup (see Fig. 1). This allows us, if desired, to solve for the orientation of the cup. Or, we can grasp the cup by approaching the cup from the top center and grasping the side wall of the cup with a parallel jaw gripper. If there are several ambiguous keypoints, as is the case for the valve in our experiments (see Fig. 3a), then all occurrences of the keypoints are labeled and considered to be of the same type.

We developed a tool, StereoLabel, to label 3D keypoints from a sequence of images taken from different viewpoints. Fig. 1, shows the tool in use. The user is shown image frames from two viewpoints. The viewpoints are picked such that the z-axes of the image frames are as close to perpendicular as possible. The image frame is defined to be z-axis forward, y down and x to the right of the image. The user labels 2D keypoints on both frames by clicking on the keypoint location in the image. In case a specific keypoint is occluded or otherwise hard to pinpoint, the user can swap out either frame with a new one.

Once corresponding keypoints are labeled, we triangulate their 3D positions in the base frame of our robot using the homogeneous direct linear transformation method [26]. We backproject the triangulated points to both frames using each frame's projection matrix, so that the user can validate that the point was appropriately placed. The user can further validate correct placement by cycling through images in the sequence by pressing a button and checking the backprojected points. In addition to the labeled keypoints, we augment the set with one additional 3D keypoint; this is the average of all the other 3D keypoints which we call the center keypoint.

Once we have all the image sequences labeled and triangulated, we can generate a dataset for training a computer vision model. Fig. 2 shows the proposed triangulation-based (stereo) and depth-based (monocular) keypoint tracking pipelines. For each frame in a sequence, we create a set of ground truth heatmaps, one heatmap for each type of keypoint. Should there be several keypoints of the same type, we pack them onto the same heatmap. We compute the 2D image coordinate for each 3D keypoint by backprojection. We place a Gaussian distribution over the 2D keypoint location computed using an RBF kernel (output of the prediction step in Fig. 2). Finally, we normalize each heatmap to have values in the range $[0, 1]$.

As there might be multiple objects in a frame, we compute 2D vector fields with vectors pointing from non-center 2D keypoints to the center keypoint. We compute one vector for output pixel having a non-zero heatmap value. With the center maps, we can associate keypoints to objects and detect multiple objects in a frame.

For the monocular version of our pipeline, we additionally compute a keypoint depth map containing the z-value of each 3D keypoint for each pixel within a fixed radius from each keypoint.

D. Learning the Keypoint Network

We use a convolutional neural network (CNN) to predict the heatmaps, along with center maps. We use CornerNet-Lite [27] as a backbone network. CornerNet-Lite is a stacked hourglass-style CNN architecture. The input is first downsampled through a series of convolutional layers and then upsampled through transposed convolutional layers in an hourglass module. Two hourglass modules are composed together.

We predict target maps with two prediction modules which take as input the output of each respective hourglass module. The prediction modules consist of three convolutional layers

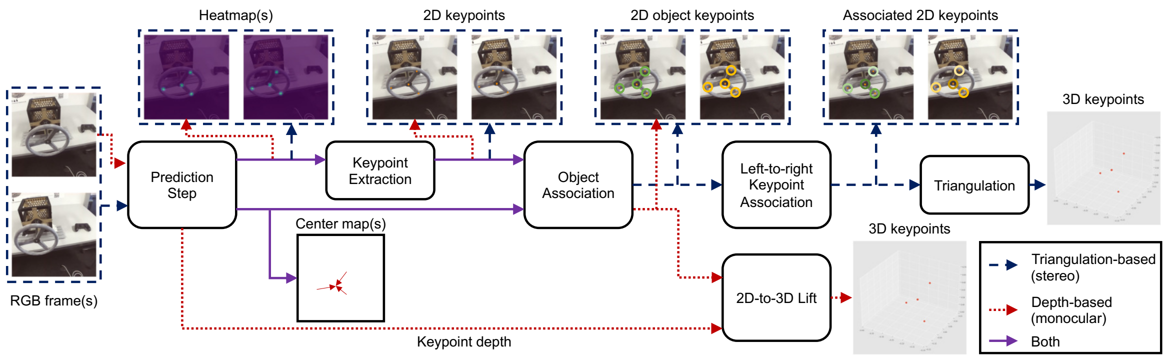


Fig. 2. The components for both of the proposed keypoint tracking pipelines.

with batch normalization, 1×1 kernels with stride 1 and relu activation functions, except for the last layer. We use sigmoid activation functions at the heatmap heads and no activation function for the center map and relu for the depth map.

The input to our network has size 511×511 pixels and the output map resolution is 64×64. We initialize the backbone network weights by pretraining on COCO [28].

We use three types of losses to train our network: a heatmap loss, a center loss and a depth loss. For the heatmap loss we use binary cross entropy:

$$\mathbf{L}_h = -\sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W} y_{cij}\log p_{cij} + (1-y_{cij})\log(1-p_{cij}). \quad (1)$$

p_{cij} is the predicted heatmap value for a keypoint of type c at output index i,j. y_{cij} is the ground truth heatmap value for keypoint map c at index i,j. C, H and W denote the amount of keypoint types, and the height and width of the output maps.

For the center loss, we simply use a smooth L1 loss:

$$\mathbf{L}_c = \sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W} \text{smooth\_L1}(\hat{\mathbf{c}}_{cij} - \mathbf{c}_{cij})\check{y}_{cij}, \quad (2)$$

where $\hat{\mathbf{c}}_{cij}$ is the center vector prediction for keypoint type c at index i,j. $\mathbf{c}_{cij}$ is the corresponding ground truth center vector. $\check{y}_{cij}$ is a binary value denoting whether the heatmap value for keypoint type c at index i,j is nonzero. The smooth L1 loss is squared below a value of 1 and linear otherwise and is applied elementwise.

To enable using a monocular camera, we additionally need an estimate of how far along the z-axis each keypoint is. To do this, we predict a pixelwise depth estimate for each keypoint type. We learn this using an L1 loss function:

$$\mathbf{L}_d = \sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W} \left\|z_{cij} - \hat{z}_{cij}\right\|_1 \check{y}_{cij}, \quad (3)$$

where z_{cij} is the ground truth depth value for keypoint c at location i,j, while $\hat{z}_{cij}$ is the corresponding estimate.

All losses are applied at both stages of the hourglass network and are combined by weighting parameters λ.:

$$L = \lambda_h(L_{h1} + L_{h2}) + \lambda_c(L_{c1} + L_{c2}) + \lambda_d(L_{d1} + L_{d2}). \quad (4)$$

L_{h1} denotes the heatmap loss at the first hourglass, L_{h2} for the second hourglass, L_{c1} the center loss for the first hourglass and so forth. When training the triangulation-based pipeline, we set the depth loss weight λ_d to 0 to disregard it entirely. We train our network using the dataset generated in Section III-C.

E. Keypoint Extraction

At runtime, we extract keypoint locations from the heatmaps by first applying a version of non-maxima supression, where we zero the non-maximum values in 5×5 regions surrounding each location. We then zero out all values below a threshold of 0.25. From each of the remaining heatmap values, we compute keypoint locations by weighing image indices by the predicted heatmap density in a 5×5 region on the unprocessed heatmap predictions centered at the maxima location.

For each non-center keypoint, we compute the object center estimate by summing the center vector with the corresponding image index. We associate each keypoint with the center keypoint closest to the keypoint's predicted center position in pixel coordinates.

F. Keypoint Association and Triangulation

After predicting and extracting keypoints in left and right image frames, we need to associate each keypoint in the left frame, to its counterpart in the right frame. To do this, we select the keypoint in the right image where $x'Fx$ is below a cutoff value of 32.0. F is the fundamental matrix derived from the camera calibration, x is the homogeneous pixel coordinates of the keypoint in the left image and x' is the homogeneous pixel coordinates of the keypoint in the right image.

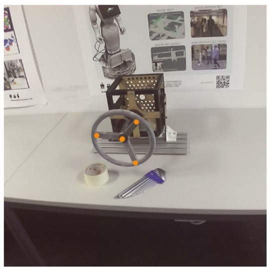
If several keypoints match, which happens when two keypoints are on the epipolar line, we shift the point by a fixed amount and pick the closest match. The fixed shift is equivalent to the difference in pixel coordinates between a point, projected onto both the left and right image frames, that is 60cm in front of the center of the left camera frame.

Finally, we triangulate the 3D location of the keypoints using the same direct linear transformation method used when creating the dataset.

G. 2D-to-3D

In the monocular version of our pipeline, we use the depth prediction, combined with the camera matrix $\mathbf{K}$ to compute the 3D point $\mathbf{X}$ corresponding to the 2D detection $\mathbf{x}$:

$$\mathbf{X} = \mathbf{K}^{-1}\mathbf{x}\hat{z}, \quad (5)$$


(a)

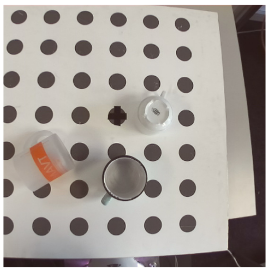

(b)

Fig. 3. (a) Valve setup showing keypoints for the value. (b) An image from the cup tracking scene.

where $\hat{z}$ is the depth estimate for keypoint $\mathbf{x}$.

IV. EXPERIMENTS

We are interested in the following questions:

- Can we use our method to quickly build up object keypoint tracking datasets?
- Can we train our keypoint tracking method on an amount of data that can be easily collected by one user?
- Is the object tracking performance good enough to enable robotic manipulation?

A. Valve: Single Object Tracking

In this experiment, we track a valve with three spokes. We define four keypoints: one at the center hub of the valve, and three at the front center points where the spokes meet the rim of the valve, shown in Fig. 3a. The three keypoints at the rim are indistinguishable from each other, and are thus considered to be of the same type and packed onto the same heatmap.

We collect 50 sequences of 30s using our data collection method, which we label using StereoLabel. The sequences differ in object arrangement, clutter, occlusion, background and lighting conditions. We split the resulting dataset into 45 sequences for training and 5 sequences for testing.

B. Label Accuracy

Our semi-automatic labeling approach has a few sources of error: synchronization between camera frames and joint encoder readings, intrinsics calibration error and hand-eye calibration error. These all result in some error in the triangulation and reprojection steps of our pipeline. Without the ground truth 3D keypoint locations, we instead manually label 2D keypoints frame-by-frame on 100 randomly sampled frames in our valve dataset and compare the human labels to ones produced by our system. This allows us to quantify how much the 2D keypoint labels drift as we observe the target object from different viewpoints. As the user does not always place the keypoints perfectly, even the human labels will have some error. We therefore establish a baseline by doing two manual frame-by-frame labeling passes to measure the variance.

C. Comparison with KeyPose

We compare our method against KeyPose [1]. KeyPose doesn't support multiple objects nor is it possible to detect keypoints on objects with multiple keypoints of the same type. We therefore can't run KeyPose on our datasets, but instead opt to evaluate our method on the KeyPose mugs dataset.

D. Cups: Multi-object Tracking

In this experiment, we seek to track up to four cups simultaneously in a scene. We collect a dataset with 100 sequences observing the cups from various viewpoints, varying the number of cups between 1 and 4 in the scene, changing the clutter, lighting conditions and background of the scene across sequences. For each scene, we randomly select between 1 and 4 cups from a set of 25 different cups. We split this dataset into 87 sequences for training and the rest for testing. We split the sequences such that 2 cups only ever occur in the test set.

V. RESULTS

A. Valve: Single Object Tracking

We timed how long it takes to label a sequence of images. Labeling a pair of valve images took us just under 15 seconds. With 50 sequences, this makes for a total of $\sim$12 minutes and 30 seconds to label all 19'507 frames in our dataset.

Table I shows the keypoint tracking performance on a held out test set. **Mean** refers to the mean error of the 3D keypoints in centimeters. **xy** is the mean error, disregarding the depth axis in the left camera frame of reference. < 3 **cm** is the percentage of measurements that were within 3 centimeters of the labeled ground truth location. **25th** and **75th** respectively denote the 25th and 75th percentiles of the combined keypoint errors. GT refers to the tracking performance using ground truth heatmaps and center maps as input with the stereo pipeline, Stereo is the stereo pipeline with a learned model, Mono is using only the left view of our stereo camera and our monocular pipeline.

Both the stereo- and depth-based pipelines perform reasonably well. For both pipelines, errors are within the range of the width of a parallel jaw gripper, and much smaller in scale than the size of the object.

1) Valve Manipulation: We deployed our keypoint tracking system on a mobile manipulation system. The goal of the experiment was to use the system to rotate the valve in Fig. 3a using the manipulator. In this case, we know the type of the valve and have a CAD model of it. We first detect the valve using keypoint tracking, and when we have detected all four keypoints (center and three spokes), we further refine the pose using ICP to match the depth readings from our camera to the

TABLE I
RESULTS FROM THE VALVE AND CUP TRACKING EXPERIMENTS.

Valve					
Method	**Mean** (cm)	**xy** (cm)	< 3 **cm**	**25th** (cm)	**75th** (cm)
GT	0.39	0.18	99.3%	0.17	0.30
Stereo	3.63	1.43	59.5%	1.29	4.26
Mono	2.99	1.06	65.0%	1.61	3.55
Cups					
Method	**Mean** (cm)	**xy** (cm)	< 3 **cm**	**25th** (cm)	**75th** (cm)
GT	1.14	0.39	97.7%	0.09	0.22
Stereo	6.71	2.24	68.5%	1.29	3.53
Mono	3.1	1.56	62.2%	1.43	4.02

object model. After refining the object pose, we command the arm to track a trajectory that rotates the valve. See the supplementary video for a successful completion of the task.

B. Label Accuracy

Comparing our generated keypoints to a manually labeled dataset, we found that the mean label difference is 6.3 pixels on average with a standard deviation of 3.4 on images with a size of 1280x720. Comparing two manually and separately labeled instances of the same datasets yield a mean difference of 2.9 pixels with a standard deviation of 1.7 pixels. While the manually labeled examples have slightly less variance, they are of the same order of magnitude for both methods.

C. Comparison with KeyPose

Table II shows results on the KeyPose mugs dataset evaluating on the unseen *mugs*_0 instance. Keypose performs slightly better. We attribute this to its more restricted problem formulation and the learned stereo image fusion employed in its network architecture.

D. Cups: Multi-object Tracking

It took us an average of 19 seconds to label a scene with 2 cups. Which makes for a total of roughly 32 minutes to label the 66'419 frames in our dataset.

Table I shows the accuracy when tracking multiple cups on a held out test set. Similarly to the valve tracking experiment, the error is larger in the depth direction. Performance of both the stereo and depth pipelines is acceptable, i.e. errors are within the width of a parallel jaw gripper. Performance is slightly worse than on the valve tracking experiment. However, we note that this is a harder task with several different objects and significantly more keypoint occlusion.

Failure modes for both pipelines include misdetecting a keypoint or associating a keypoint with the wrong object. Failure modes of the stereo pipeline also include misassociating keypoints from left-to-right and a bad triangulation due to slightly misdetected 2D keypoints. Additionally, both approaches fail when two keypoints of the same type align, either from the same or different objects, and occlude each other. In such cases, the keypoints will get detected as one and the center prediction might point toward either object in the case of multiple objects.

Table III shows how long each step of our stereo pipeline takes on average for our implementation. The measurements were made on a computer with an Nvidia RTX 2080 GPU and AMD EPYC 7742 CPU.

TABLE II
OUR MONOCULAR PIPELINE ON THE KEYPOSE MUGS DATASET.

Method	MAE (cm)	< 2 cm
KeyPose	1.6	78.6%
Ours monocular	2.0	66.4%
Ours Stereo	1.9	69.7%

TABLE III
TIME SPENT ON EACH STEP OF THE STEREO PIPELINE.

Stage	Mean time (ms)
Prediction	32.9
Keypoint extraction	6.8
Object association	0.62
Left-to-right association	0.1
Triangulation	0.2

VI. DISCUSSION AND CONCLUSIONS

In this paper, we presented a method to quickly collect and label object keypoint tracking datasets and a system that learns to recover the labels at runtime on unseen examples. We showed that we fully rely on calibration to avoid having to place markers in the environment. In experiments, we showed that we can generate accurate object keypoint labels much quicker than using a 2D labeling approach, while also annotating the z-dimension and without having to create segmentation maps. We showed that our presented system is able to detect keypoints on multiple objects simultaneously at real-time rates, using the produced datasets. We showed that it can be successfully used as part of a system to solve real world manipulation tasks.

While we presented two different keypoint tracking algorithms, the actual keypoint and object detection algorithm can be replaced with any other pipeline, as long as the labels can be derived from our data collection method. Additional information about the objects could also be used to further improve the estimated keypoints. The Perspective-n-Point algorithm could be used for known objects or a category-level object model could be fitted to the keypoint detections to further improve them, similar to what is used in [29]. Additional computation could be traded for accuracy by predicting keypoint heatmaps at a higher resolution. The 64x64 pixel output resolution we used is quite limiting.

One weakness of our data collection method, is that it relies on measurement timestamps from different sensors. On our platform, these are not hardware synchronized. Some cameras can be tightly synchronized, while triggering boards such as the VersaVIS board exist [30] which are able to synchronize several cameras and IMUs. Extending these to cover other types of sensors, such as joint encoders, would improve the usability and accuracy of our proposed method.

Finally, when collecting our data, we manually guide the robot into different viewpoints. This could be automated. Furthermore, the robot could semi-autonomously improve upon an initial learned model using an active learning type approach. Different models and viewpoints could be used to bootstrap a dataset, similar to what is done in [11].

VII. ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under project PILOTING No H2020-ICT-2019-2 871542 and under grant agreement No 101017008 (HARMONY).

REFERENCES

[1] X. Liu, R. Jonschkowski, A. Angelova, and K. Konolige, "KeyPose: Multi-view 3D labeling and keypoint estimation for transparent objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 602–11 610.

[2] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake, "LabelFusion: A pipeline for generating ground truth labels for real RGBD data of cluttered scenes," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3235–3242.

[3] 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*, 2020, pp. 11 632–11 641.

[4] W. Gao and R. Tedrake, "kPAM-SC: Generalizable manipulation planning using keypoint affordance and shape completion," *arXiv preprint arXiv:1909.06980*, 2019.

[5] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 international conference on advanced robotics (ICAR)*. IEEE, 2015, pp. 510–517.

[6] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *Proceedings of Robotics: Science and Systems*, 2018.

[7] T. Grenzdörffer, M. Günther, and J. Hertzberg, "YCB-M: a multi-camera RGB-D dataset for object recognition and 6DoF pose estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3650–3656.

[8] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," *arXiv preprint arXiv:1809.10790*, 2018.

[9] D. Stumpf, S. Krauß, G. Reis, O. Wasenmüller, and D. Stricker, "SALT: A semi-automatic labeling tool for RGB-D video sequences," in *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP*, 2021, pp. 595–603.

[10] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut" - Interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.

[11] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1145–1153.

[12] Y. Yao, Y. Jafarian, and H. S. Park, "MONET: Multiview semi-supervised keypoint detection via epipolar divergence," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 753–762.

[13] 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*, 2018, pp. 292–301.

[14] K. Park, T. Patten, and M. Vincze, "Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7668–7677.

[15] S. Zakharov, I. Shugurov, and S. Ilic, "DPOD: 6D pose object detector and refiner," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1941–1950.

[16] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise voting network for 6DoF pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570.

[17] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6D object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3385–3394.

[18] S. Suwajanakorn, N. Snavely, J. J. Tompson, and M. Norouzi, "Discovery of latent 3D keypoints via end-to-end geometric reasoning," in *Advances in Neural Information Processing Systems*, 2018, pp. 2059–2070.

[19] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu, "6-PACK: Category-level 6D pose tracker with anchor-based keypoints," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 059–10 066.

[20] M. Vecerik, J.-B. Regli, O. Sushkov, D. Barker, R. Pevceviciute, T. Rothörl, C. Schuster, R. Hadsell, L. Agapito, and J. Scholz, "S3K: Self-supervised semantic keypoints for robotic manipulation via multi-view consistency," *arXiv preprint arXiv:2009.14711*, 2020.

[21] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6D object pose and size estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2642–2651.

[22] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "kPAM: Keypoint affordances for category-level robotic manipulation," in *International Symposium on Robotics Research*, 2019.

[23] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei, "Integral human pose regression," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 529–545.

[24] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1360–1367.

[25] F. Furrer, M. Fehr, T. Novkovic, H. Sommer, I. Gilitschenski, and R. Siegwart, *Evaluation of Combined Time-Offset Estimation and Hand-Eye Calibration on Robotic Datasets*. Cham: Springer International Publishing, 2017.

[26] A. M. Andrew, "Multiple view geometry in computer vision," *Kybernetes*, 2001.

[27] H. Law, Y. Teng, O. Russakovsky, and J. Deng, "CornerNet-Lite: Efficient keypoint based object detection," *arXiv preprint arXiv:1904.08900*, 2019.

[28] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[29] H. Yang, C. Doran, and J.-J. Slotine, "Dynamical pose estimation," *arXiv preprint arXiv:2103.06182*, 2021.

[30] F. Tschopp, M. Riner, M. Fehr, L. Bernreiter, F. Furrer, T. Novkovic, A. Pfrunder, C. Cadena, R. Siegwart, and J. Nieto, "VersaVIS—an open versatile multi-camera visual-inertial sensor suite," *Sensors*, vol. 20, no. 5, p. 1439, 2020.