S3K: Self-Supervised Semantic Keypoints for Robotic Manipulation via Multi-View Consistency

Mel Vecerik
University College London, DeepMind
mel.vecerik.18@ucl.ac.uk, vec@google.com

Jean-Baptiste Regli
DeepMind
jbregli@google.com

Oleg Sushkov
DeepMind
sushkov@google.com

David Barker
DeepMind
davebarker@google.com

Rugile Pevceviciute
DeepMind
rugile@google.com

Thomas Rothörl
DeepMind
tcr@google.com

Christopher Schuster
DeepMind
cschuster@google.com

Raia Hadsell
DeepMind
raia@google.com

Lourdes Agapito
University College London
l.agapito@cs.ucl.ac.uk

Jonathan Scholz
DeepMind
jscholz@google.com

Abstract:
A robot's ability to act is fundamentally constrained by what it can perceive. Many existing approaches to visual representation learning utilize general-purpose training criteria, *e.g.* image reconstruction, smoothness in latent space, or usefulness for control, or else make use of large datasets annotated with specific features (bounding boxes, segmentations, etc.). However, both approaches often struggle to capture the fine-detail required for precision tasks on specific objects, *e.g.* grasping and mating a plug and socket. We argue that these difficulties arise from a lack of *geometric structure* in these models. In this work we advocate *semantic 3D keypoints* as a visual representation, and present a self-supervised training objective that can allow instance or category-level keypoints to be trained to 1-5 millimeter-accuracy with minimal supervision. Furthermore, unlike local texture-based approaches, our model integrates contextual information from a large area and is therefore robust to occlusion, noise, and lack of discernible texture. We demonstrate that this ability to locate *semantic keypoints* enables high level scripting of human understandable behaviours. Finally we show that these keypoints provide a good way to define reward functions for reinforcement learning and are a good representation for training agents.

Keywords: Semantic keypoints, self-supervised learning, robot manipulation

1 Introduction

It is well understood that manipulating objects precisely requires (at least) an implicit understanding of the 3D relationships between the robot and objects in space. Most existing visual representations achieve this either by full supervision [1, 2, 3, 4], dense self-supervision [5, 6], or else lack 3D structure entirely [7, 8, 9, 10, 11, 12]. In this paper we explore a middle ground: our goal is to learn an explicit 3D representation that captures *task-relevant* object geometry, without requiring depth-sensing or full supervision. We argue that these requirements are critical in order to achieve generalizable manipulation behavior in an inherently 3D world, while being practical to train.

Our approach utilizes multiple camera views to train a model to extract keypoints located on relevant object parts. This representation offers two key benefits: 1) the use of multi-view geometry constrains the hypothesis space to geometrically meaningful features, and 2) it provides an intuitive way for humans to tell robots what to attend to, *e.g.* by clicking on an image.

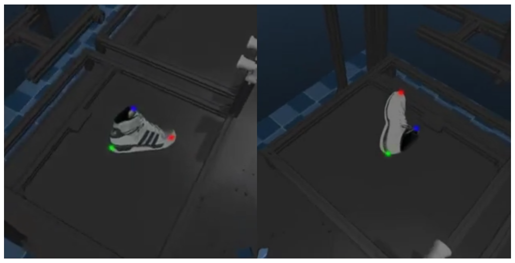
Figure 1: Exploring the generalization ability of the keypoint model.

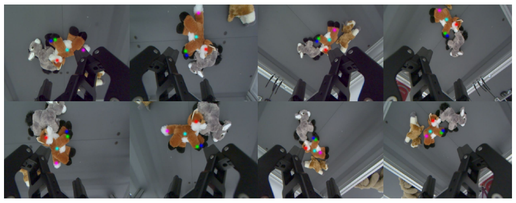
Figure 2: Keypoint detections of unseen configurations of the plush objects.

However, this approach raises the question of how to obtain sufficient annotations without a tiresome labeling process. S3K is a novel semi-supervised approach which allows us to use a small number of 2D annotations to specify what to focus on, along with a large corpus of unlabeled images to generalize to a wide range of conditions.

Our central finding is that the self-supervised component of the model is capable of extracting 3D structure from data even with an extreme scarcity of ground-truth labels. This allows a keypoint detector to be trained to a high accuracy using a surprisingly small number of annotated images. In addition, our approach naturally permits keypoints not to be tied to surface points – they can be inside objects or correspond to no physical location at all as long as they are semantically and geometrically consistent. We evaluate this model across a set of simulated and real-robot manipulation tasks, involving both rigid and deformable objects. Videos and further materials available at sites.google.com/view/2020-s3k.

2 Related work

The overarching goal of this paper is to enable the creation of new behaviours through better representations. Finn et al. [8] showed that policies can be trained on top of a spatial-softmax operation which allows a neural network to extract 2D image *coordinates* from convolutional feature maps. This operator, however offers no explicit reasoning in 3D space and in practice feature maps can be noisy and waste representational capacity on task-irrelevant detail, especially in non-planar tasks. This effect can be reduced *e.g.* by using triplet or contrastive losses [13], or by using self-supervision across multiple input modalities [14]. TCN [15] achieves this in a multi-view setting similar to ours, but does not exploit camera geometry. Recently, several works use invariance to cropping as a way to constrain representations [16, 17, 18]. However, all of these representations lack 3D awareness and are therefore hard to interpret within the setting of a robot controller.

In 3-dimensions, keypoints are a foundational and common representation that can be used to extract higher-level object features such as 6D-pose [3, 4]. Alternatively they can be used directly for robot manipulation as in [19] which defines an optimization problem (*e.g.* shoe reorientation) in terms of 3D keypoints. [4] showed that keypoints can significantly improve 6-DOF tracking performance on transparent objects, which is known to be a challenging problem for both template and direct-regression methods [20]. [19] and [4] also described a geometry-based mechanism to allow label propagation across time, which we employed in our grasp experiments below. However, this treatment requires the scene to be static, which precludes training from dynamic interactions.

Other related methods utilize keypoints internally to obtain 6D pose estimates, but do not require explicit supervision on the keypoints [6, 21, 22]. Representations constructed for camera localization are biased to discard moving objects which makes them unsuitable for most manipulation tasks. A more robotics-relevant approach was shown in [23] which uses reconstructions within episodes to learn a keypoint detector in an unsupervised way. However, rich and moving backgrounds quickly use up model capacity which limits their ability to precisely represent small objects or specific object parts.

Keypoints are also very common in human and hand pose tracking [24]. [5] enforces a 3D-aware latent representation by predicting an image from one viewpoint given an image from another. However, the image reconstruction loss makes it dependent on high quality background subtraction. A different approach was demonstrated by [25] where consistency across multiple cameras is used to predict dynamic poses of skiers. This work is similar to ours however they do not use calibrated

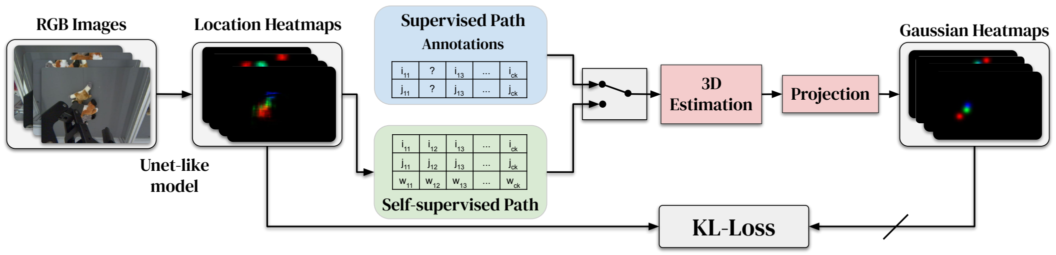
RGB Images
Location Heatmaps
Unet-like model
Supervised Path
Annotations
i_{11} ? i_{13} ... i_{ck}
j_{11} ? j_{13} ... j_{ck}
i_{11} i_{12} i_{13} ... i_{ck}
j_{11} j_{12} j_{13} ... j_{ck}
w_{11} w_{12} w_{13} ... w_{ck}
Self-supervised Path
3D Estimation
Projection
Gaussian Heatmaps
KL-Loss

Figure 3: Diagram of the learning setup showing the supervised and the self-supervised paths. The model predicts location heatmaps for each camera and keypoint. To learn from annotations we need the keypoint to be labelled in at least 2 out of 4 cameras. Otherwise we use the coordinates and uncertainties from the model to estimate the keypoint locations instead. These 3D locations are then projected back to the camera frames and provide a learning target for the model via a KL-loss.

cameras, which restricts them to predicting *normalized* poses. This is an important limitation for us because in our tasks the absolute scale is important.

Lastly, the goal of our work is also closely related to [26] where a mapping is learned from camera images to a per-pixel dense embedding. This is done by considering pixels which represent the same spatial location from different views and considering their embeddings as positives in a contrastive loss. 3D keypoints can be constructed by locating a specific embedding in pixel space and combining it with depth information. Further in [27] demonstrates that these methods can work from dynamic depth maps by using simultaneous image acquisitions from different views, instead of different time-points. This is the same approach we pursue here, but we triangulate depth from multi-view geometry rather than requiring a calibrated depth-sensor. A limiting factor of these approaches is that they rely on having quality depth maps to create the image correspondences and most current depth sensors struggle to provide good data for thin or highly reflective objects. In addition, depth-sensors only provide correspondences on the object surface, unlike our method and [19], which we believe is important in order to handle occlusions or arbitrary rotations.

3 Method

There are two core components of S3K: (1) a neural-network which is trained to detect 2D keypoints for all individual scene views, and (2) a 3D estimation layer which consumes the 2D predictions and generates an estimate of the 3D location of each keypoint. At a high-level, the network is trained to achieve a spatial consensus among the 2D predictions by projecting the 3D estimate *back* to each camera view and penalizing divergence. This structure allows the network to be trained from a mixture of labelled and unlabelled data – labels are only required to ground the model's ability to target specific points, beyond which consensus is sufficient to drive further generalization. In this section we detail the operation of both training pathways, and a simple heuristic for handling missing or uncertain predictions. Fig. 3 depicts the structure of these components.

3.1 Supervised keypoint training

In the supervised regime we have access to 2D labels of a single point in space from multiple perspectives. Given the camera calibration we can compute the 3D location $\mathbf{x}_k$ of all K keypoints even when annotations from all cameras are not available. Similarly, we can project any 3D location back onto the image planes. We write the image coordinates of the back-projected keypoint k into the camera image c as i_{ck}, j_{ck} and we construct a Gaussian image centered at this location with width σ. This creates a heat-map of the following form: $H_{ck}(ij) \propto \exp\left((-(i_{ck}-i)^2-(i_{ck}-j)^2)/(2\sigma^2)\right)$ where i and j are coordinates in the heatmap.

To define the loss, we write the images as I_c, the model as f, and the model parameters as θ. Next we obtain the model's prediction $P_{ck} = \text{softmax}(f(I_c,\theta)_k)$ where the softmax is performed separately for each keypoint and image. We emphasize that the *same* network is applied to each viewpoint to obtain the heatmaps for all keypoints. In the loss, we sum the D_{KL} terms between the heat-map $H_{ck}(ij)$ and the model prediction P_{ck} for all C camera images, and all K keypoints. From this we can write the final supervised loss as:

$$\mathcal{L}_{sup} = \sum_{c}^{C}\sum_{k}^{K} D_{\mathrm{KL}}\left(H_{ck} || softmax(f(I_c, \theta)_k)\right) \tag{1}$$

3.2 Self-supervised keypoint training

In the self-supervised case we do not have explicit labels, but we do assume access to multiple simultaneous images of the scene. Note that in the supervised case we did not directly train on the 2D annotations, but rather estimated a 3D location and re-projected this hypothesis to each camera to obtain a target heatmap. The same procedure can be followed with any 3D location, so we can use a bootstrapped estimate rather than an annotation. This allows the model to self-supervise from its own predictions in a 3D-consistent way. This corresponds to the bottom pathway in Fig. 3.

To estimate the 3D location we generate the heat-map predictions $P_{ck} = \mathrm{softmax}(f(I_c, \theta)_k)$ as in the supervised case. The mean of P_{ck} corresponds to an image position, from which a direction can be computed via calibration similarly to [4, 28]. This creates a ray for each camera in the direction of its estimate. Assuming that every keypoint appears on the scene exactly once, we define our keypoint position estimate $\widetilde{\mathbf{x}}_k$, as a point for which the sum of squares of distances from these rays is minimized. This defines a quadratic cost for $\widetilde{\mathbf{x}}_k$ and therefore can be analytically solved (Eq. 2),

3.2.1 Detection weight

It is not always desirable to incorporate all predictions from all cameras, e.g. due to certain keypoints dropping out of view or becoming occluded. Therefore we incorporated a weight factor w_{ck} to reflect confidence of the heatmap predictions. This weight factor is used to multiply the quadratic loss associated with this detection. Further details about this weight factor are provided in Appendix B. The equation for self-supervised keypoint estimation is obtained by solving the weighted least-squares problem discussed in Section 3.2.

If we define the location of camera c as $\mathbf{a}_c$ and the normalized direction of the ray from camera c through keypoint k as $\hat{\mathbf{d}}_{ck}$, the following equation yields the estimated 3D keypoint location:

$$\widetilde{\mathbf{x}}_k = \left(\sum_{c}^{C} \mathbf{I} - \hat{\mathbf{d}}_{ck}\hat{\mathbf{d}}_{ck}^{\intercal}\right)^{-1} \left(\sum_{c}^{C} w_{ck}(I - \hat{\mathbf{d}}_{ck}\hat{\mathbf{d}}_{ck}^{\intercal})\mathbf{a}_c\right) \tag{2}$$

This is treated as a label and is not back propagated through. The self-supervised loss is defined as:

$$\mathcal{L}_{unsup} = \sum_{c}^{C}\sum_{k}^{K} D_{\mathrm{KL}}\left(\widetilde{H}_{ck} \middle|\middle| softmax(f(I_c, \theta)_k)\right) \tag{3}$$

Where $\widetilde{H}_{ck}$ is created from our estimated keypoints locations $\widetilde{\mathbf{x}}_k$ rather the annotated locations $\mathbf{x}_k$. In the final loss we also include L2 regularization on all non-bias parameters.

$$\mathcal{L}_{total} = \mathcal{L}_{sup} + \alpha\mathcal{L}_{unsup} + \lambda||\theta_{nonbias}||^2 \tag{4}$$

4 Investigating the model

In our experiments we aim to answer the following questions: (1) What is the relative value of labelled vs. unlabelled data? (2) Can the self-supervision mechanism be used to provide robustness to domain shift in the absence of further labels? (3) Is the model sufficiently accurate and stable to use for real-robot control, and how does this compare to a non-geometric alternative?

4.1 Value of labelled vs. unlabelled data

Unlabelled data is typically less useful for training than labeled data, so one of our key objectives is to evaluate the extent to which this holds for S3K. For this experiment we use a simulation, because it allows us to generate large amounts of well controlled data.

Dataset. To gather data we created a simple simulated environment similar to our physical robotic cell using MuJoCo[29] and added a realistic 3D scanned box of a boardgame. The keypoints were annotated at 3 corners of the box. For each sample we randomized the box location and orientation,

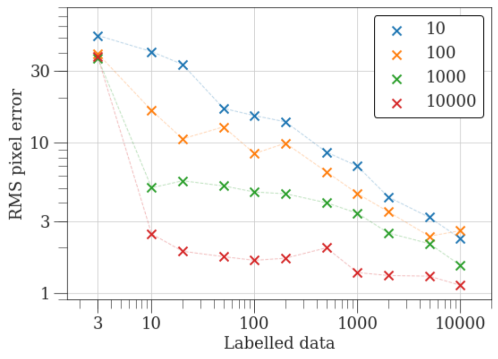

Figure 4: Comparing effects of dataset sizes on keypoint detection error on a simulated task. Each curve corresponds to a number of *unlabelled* datapoints.

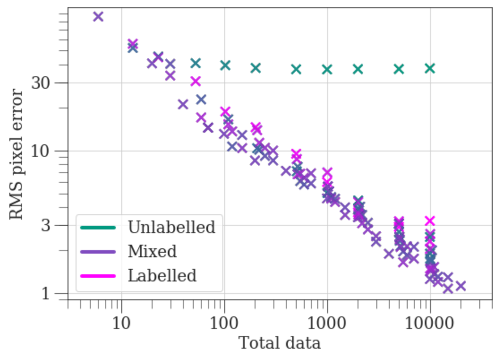

Figure 5: Scaling of prediction error with total amount of data. Color denotes the ratio between the labelled and unlabelled data.

as well as directions of 4 cameras from which we render the scene. All images were 400px by 400px. We generated a set of datasets of labelled and unlabelled data of various sizes. The unlabelled datasets included the camera positions, but not the keypoint positions. For each experiment we used one labelled and one unlabelled dataset and trained the model for 150k batches. RMS pixel error on a withheld set of samples is used as the evaluation metric and is averaged over 3 seeds.

Results and discussion. Fig. 4 shows that with 3 labelled samples the model cannot successfully learn, but that 10 labelled samples (*i.e.* 40 images) is sufficient to obtain a good model if given enough unlabelled data. There is also a steady improvement in the performance as the amount of unlabelled data increases, especially in cases where little labelled data is provided.

Another way to visualize these results is to plot the total amount of data on the x-axis and use color to denote the ratio between the amounts of labelled and unlabelled data as in Fig. 5. Here we see a clear trend in accuracy as a function of data, *regardless of the kind of data*. This result suggests that beyond a small threshold, labelled and unlabelled data are equally valuable. A careful observer may notice that almost fully labelled datasets perform worse than mixed ones. This is an artifact of the training which arises because the two losses are combined with $\alpha = 0.5$ and the model overfits to the unlabelled data.

4.2 Domain shift experiments

The scenario that we are exploring here is one in which the environment distribution changes after the robot was deployed. To simulate this we assume that we have annotations of an initial scene. This scene consists of our object of interest as well some other task-irrelevant distractor objects. The distractor objects change over time, but only unlabelled data from this process is available. After the distractors are switched multiple times, we evaluate the ability of the model to predict the keypoint locations with new distractor objects present.

Dataset. This dataset is based on the one described in Section 4.1 with extra objects of similar size and textures. We call the scenario described above the *start mode* annotation. Our baseline is comparing to a model which has access to labelled data spread throughout the whole timeline instead of just the beginning - we call this the *full mode* annotation.

Results and discussion. Fig. 6 shows that having access to the *full* data is beneficial. This gap consistently narrows as we increase the amount of unlabelled data. Also we can see that the *critical number* of labelled data has shifted to about 50 for this harder dataset.

We can draw conclusions from this experiment by considering Fig. 6, which looks at the case of changing the amount of labelled data for both *full mode* and *start mode* distributions. Each line represents a fixed amount of unlabelled data. We see that when we have little unlabelled data (*i.e.* blue and orange) there is a big performance gap between the *start* and *full* versions especially as the amount of labelled data increases towards the right. This means that the model is significantly overfitting to the initial distractors. We can contrast this to the behaviour of the purple curve which

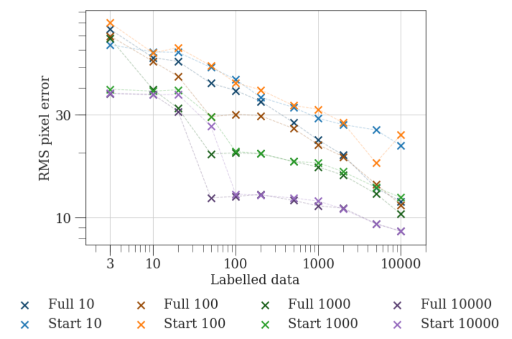

Figure 6: Using unsupervised data to gain robustness to domain shift.

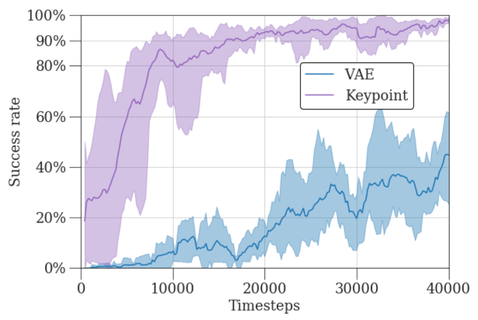

Figure 7: Success rates of agents on an audio cable insertion task comparing using different vision features for RL.

represents having much more unlabelled data. Here the gap between *start* and *full* is minimal showing that the model is much less sensitive to the choice of distribution for the labelled data. This means that the unlabelled data can be used to address the issue of domain shift.

4.3 Within class generalization

So far we have focused on tracking keypoints grounded on specific object instances. However, nothing about this model or architecture precludes learning category-level semantics. In this section we demonstrate that this model is indeed capable of tracking consistent parts of a single *class* of objects. For this purpose we created a set of datasets in which we annotated 16 shoes at their *front*, *tongue* and *heel*. Examples of images from these datasets can be seen in Fig. 1.

	Train	**Test**
1	0.78	26.99
2	1.59	14.65
3	1.73	16.22
6	1.65	15.05
8	1.82	11.75
12	1.74	10.77

Table 1: RMS pixel error on the shoe keypoints as a function of the number of unique training examples.

This time in addition to randomizing the object position at test-time, we also evaluate on 4 held-out shoes not seen during training. Building on the domain-shift experiment, we compare models that had access to ground-truth labels on varying numbers of shoes during training, from 1-12. In Table 1 we see that this model is able to generalize reasonably well to unseen shoes from as few as 2 training instances, and that accuracy increases with additional instances as expected. Although the gap between training and test errors is large, in a real setting a 10px accuracy would be sufficient for a reliable grasp of either part of the shoe. We encourage the reader to view the supplementary video for qualitative results.

5 Keypoints for scripted behaviours

Unlike unstructured visual representations, keypoints are human understandable and provide actionable representation for robotic controllers. To demonstrate this, we put 3 flexible plush toys into the robotic cell and trained the keypoint model to track the head, front paws, body and tail of a plush fox. Pictures of this scene can be seen in Fig. 2. The other 2 toys acted as distractors. We arranged the toys into 10 different arrangements and collected 3k timesteps (i.e. 12.5 minutes) from each. We manually labelled a single frame in each arrangement and used the stationarity of the scene to propagate the labels into all of the timesteps similarly as in [19]. 7 of these arrangements were used for training and 3 for testing, which gave us 21k training timesteps in total. The model is able to track all of the keypoints well, except for the tip of the tail which varied significantly between views and wasn't always visible. The front and back paws are also sometimes confused especially in scenarios where the head and tail are occluded by the gripper in multiple views.

To demonstrate the accuracy of the model we scripted a motor primitive that grasped at a specified 3D location. We demonstrate this behavior by consistently grasping the fox by the front paw. Note that rigid object 6D pose tracking would not be applicable here due to the deformability of the toy. The direction of the grasp is deduced by considering the *body* keypoint as well. Since our keypoint is *inside* the object there is no need for additional processing between the keypoint and grasp point as would be case if the keypoint was on the surface. This behaviour was repeatedly run and the

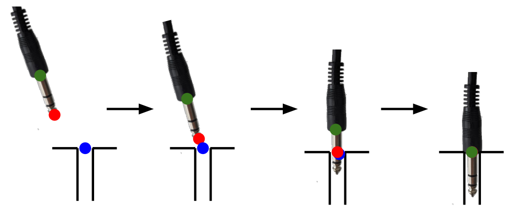
Figure 8: Diagram of keypoints annotation on the cable.

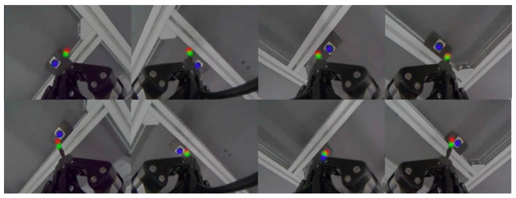
Figure 9: Model detections for two points in time. Each row is a single time-step.

main failure point was when the right paw was not immediately graspable. To achieve a more robust controller we could either improve the detections by collecting additional unlabelled data, or improve the controller itself by using the features for RL. Both of these are used in the next section.

This shows that this model is capable of generalizing from a small number of labelled scenes to novel arrangements. The right paw is a small target within the workspace, yet it could be located reliably enough to perform the physical task. This demonstrates a workflow where tasks requiring precision and generalization can be solved with only a small amount of human labour.

6 Keypoints for reinforcement learning

Our ultimate objective is to build agents that can be quickly taught to perform complex manipulation skills. So far we have focused purely on how the 3D keypoint representation allows for tracking and scripted control, but for our final experiment we sought to evaluate the representation on a task that would be difficult or impossible to script by hand. Our main goal was to determine whether the 3D keypoint model would hold up in a closed loop RL setting, and if so, whether it could outperform a more general-purpose visual representation.

There are several reasons it might perform poorly, *e.g.* instability in the model prediction, or bias to an overly restrictive feature-set. However, by its nature RL can learn to be robust to multiple forms of noise, and strongly benefits from representations that allow generalization and data-efficiency.

6.1 Task

The task we selected was a deformable cable insertion problem, depicted in Fig. **??**. In this task the robot is holding a connector by the cable, and the goal is to insert it into a vertical socket. The cable has sufficient flex to easily bend under gravity alone, and does not straighten to a consistent shape when removed. To randomize the environment between episodes we deliberately bend the cable using a scripted motion. Actions were 6-dimensional Cartesian velocity commands at 4Hz and observations included gripper pose and velocity in addition to visual features.

This task is challenging to script because the cable deforms in complex ways when making contact with objects, and the relationship between robot actions and the cable state is highly non-linear. Clearly two keypoints are insufficient to describe the full state of the cable, so this task also helps to evaluate whether the benefit of incorporating 3D structure is worth the cost of restricting the agent's attention to a finite number of points. If the cable contour or other scene information is important, one might expect a VAE to outperform a keypoint model on this task.

6.2 Agent

We use a setup similar to EDRIAD [7], *i.e.* a DPG [30] agent with demonstrations and keypoint locations as the visual features. We used 51 successful episodes as demonstrations to accelerate training. As keypoints we annotated the plug tip, plug base and socket positions as shown in Fig. 8. As the plug tip inserts into the socket, we start tracking the socket opening instead, as we cannot consistently track the tip anymore[1]. This exploits an important feature of keypoints which is that they do not need to be attached to a specific point on the surface or a physical location as long as they are multi-view consistent.

[1]As discussed in Section 3, the choice of what to track is implied by the labels – *i.e.* we labeled the plug-tip point at the socket opening after insertion.

As a baseline we trained an equivalent agent with a β-VAE[31] as the vision model. It embedded all 4 camera views into a single 10 dimensional latent embedding. Both models were trained on the same unlabelled data, but the keypoint model received extra supervision from the labels. Even this baseline agent used keypoint-distance between the plug and socket for rewards, although any reward mechanism e.g. electrical-connectivity could be substituted.

We gathered $\approx$ 3k timesteps for both vision models. For the keypoint model we labelled 120 of them for the training and additional 36 for testing to enable hyperparameter tuning using vizier [32]. We defined the task success as the moment when the sum of the distances between all 3 keypoints is below 2.5cm as this means that the plug is almost fully inserted. This distance minimizes false-negatives while preventing premature termination. For safety, the episodes are also terminated if the agent moves far away from the socket or the episode takes too long (over 10s).

6.3 Results and discussion

Each experiment was run for 40000 time-steps which corresponds to about 3 hours of training. We ran 5 runs for each setup and averaged the results. Within each run we computed a rolling average over 2000 timesteps. As we can see in Fig. 7, the keypoint-based agent significantly outperformed ones trained with VAE features. Error bars show min and max across seeds after the rolling average. By the end of the experiment the keypoint agents reliably and repeatedly attempt insertion and final performance increases (above 95%) correspond to their ability to succeed within the time limit. VAE based agents never reach this performance even when trained for a significantly longer time as their training slows down after they reach success rate above 60%. We believe this is because the VAE representation lacks *the precision* to master this task. This shows that the keypoint representation is superior for these use-cases.

During the agent training, we would occasionally see keypoint mis-detections, however the chance of all keypoints collapsing to a small range was sufficiently small that we did not observe any false-positives. This improves upon previous work where an ensemble had to be used to alleviate the issue of false positives [7]. These agents are capable of handling the noise from the model as well as implicitly reason about the deformations through the keypoint locations. These results show that keypoints can act as a good *summary* of the visual scene, and can be used as visual features for robotic tasks to avoid having to learn from raw images.

7 Conclusions

In this paper we have built upon previous work in representations for reinforcement learning [7], supervised keypoint learning [4] and geometry-based unsupervised keypoint learning [5]. We have introduced a new self-supervised loss that, when combined with a supervised loss on a small number of labelled samples, can provide a robust detector for semantic 3D keypoints. We call this setup S3K and its main contribution comes from considering multi-view geometry as a source of self-supervision for keypoint based models and showing its applicability to robotic tasks.

We have conducted experiments on simulated data to investigate how S3K scales with different amounts of labelled and unlabelled data. In particular, we showed that, given a small but sufficient set of labelled examples, further performance scales with the total amount of data, not the total labelled data. We have further demonstrated how unlabelled data combined with S3K can be used to counteract effects of domain shift and shown its ability to generalize across samples from the same category.

To demonstrate applicability on real robot scenarios we used the model outputs to script a policy to lift a plush toy fox by its front right paw in the presence of distractor objects. In addition we apply S3K to reinforcement learning by using it to define a reward function and to provide visual features for an agent to learn a policy to insert a flexible audio cable into a socket.

We believe that our self-supervised keypoint detection via multiview consistency could be applied to many related fields as a way to reduce supervision. We can see a trend in the field as we move away from supervised models towards models which require less and less supervision. S3K dramatically reduces the need for supervision, but we believe that in the future we will be able to develop related methods which will remove the need for any explicit human grounding in terms of keypoints completely, but instead will be supervised purely via the task or demonstrations.

8 Acknowledgments

We would like to thank Giulia Vezzani, Michael Bloesch, and Martin Riedmiller for helpful reviews as well as Serkan Cabi, Misha Denil for fruitful discussions. Further we would like to thank Francesco Nori for making this work possible. We would also like to thank Federico Casarini, Stefano Salicety and Nathan Batchelor for help with the physical setup of the cell and Toby Turner and Dante Pomells for helping us setup the experiments when we couldn't be physically present.

References

[1] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[2] M. Rad and V. Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3836, 2017.

[3] 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*, pages 4561–4570, 2019.

[4] 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*, pages 11602–11610, 2020.

[5] H. Rhodin, M. Salzmann, and P. Fua. Unsupervised geometry-aware representation learning for 3d human pose estimation. In *ECCV*, 2018.

[6] J. Tang, R. Ambrus, V. Guizilini, S. Pillai, H. Kim, and A. Gaidon. Self-supervised 3d keypoint learning for ego-motion estimation. *arXiv*, 2019.

[7] M. Vecerik, O. Sushkov, D. Barker, T. Rothörl, T. Hester, and J. Scholz. A practical approach to insertion with variable socket position using deep reinforcement learning. *2019 International Conference on Robotics and Automation (ICRA)*, pages 754–760, 2019.

[8] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016.

[9] L. Yen-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin. Learning to see before learning to act: Visual pre-training for manipulation. 2020.

[10] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

[11] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL http://arxiv.org/abs/1312.6114.

[12] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[13] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(36):1109–1135, 2010. URL http://jmlr.org/papers/v11/chechik10a.html.

[14] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950, 2019.

[15] P. Sermanet, C. Lynch, J. Hsu, and S. Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. *arXiv preprint arXiv:1704.06888*, 2017.

[16] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. arXiv:2004.14990, 2020.

[17] M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119*, 2020. arXiv:2003.06417.

[18] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *ArXiv*, abs/2004.13649, 2020.

[19] L. Manuelli, W. Gao, P. R. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. *ArXiv*, abs/1903.06684, 2019.

[20] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[21] T. Cieslewski, M. Bloesch, and D. Scaramuzza. Matching features without descriptors: Implicitly matched interest points (imips). *arXiv preprint arXiv:1811.10681*, 2018.

[22] 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 31*, pages 2059–2070. Curran Associates, Inc., 2018.

[23] T. Jakab, A. Gupta, H. Bilen, and A. Vedaldi. Unsupervised learning of object landmarks through conditional image generation. *arXiv*, 2018.

[24] Y. Chen, Y. Tian, and M. He. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192:102897, 2020. ISSN 1077-3142. doi:https://doi.org/10.1016/j.cviu.2019.102897.

[25] H. Rhodin, J. Spörri, I. Katircioglu, V. Constantin, F. Meyer, E. Müller, M. Salzmann, and P. Fua. Learning monocular 3d human pose estimation from multi-view images. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8437–8446, 2018.

[26] P. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *Conference on Robot Learning*, 2018.

[27] P. Florence, L. Manuelli, and R. Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492499, Apr 2020. ISSN 2377-3774. doi:10.1109/lra.2019.2956365. URL http://dx.doi.org/10.1109/LRA.2019.2956365.

[28] T. Hodan, D. Baráth, and J. Matas. Epos: Estimating 6d pose of objects with symmetries. *ArXiv*, abs/2004.00605, 2020.

[29] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Proc. of IROS*, pages 5026–5033, 2012.

[30] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[31] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[32] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1487–1495, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4887-4. doi:10.1145/3097983.3098043.

[33] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *LNCS*, volume 9351, pages 234–241, 10 2015. ISBN 978-3-319-24573-7. doi:10.1007/978-3-319-24574-4_28.

[34] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Appendix A Network architecture

We follow standard practice in using a convolutional network to process images and obtain high-level features. However, fundamentally we care about pixel-accurate detection – *where* these keypoints are in the image. U-net's architecture [33] is a good fit for this problem, by allowing both a deep convolutional downsampling path for large receptive field as well as a resolution-preserving path for precision.

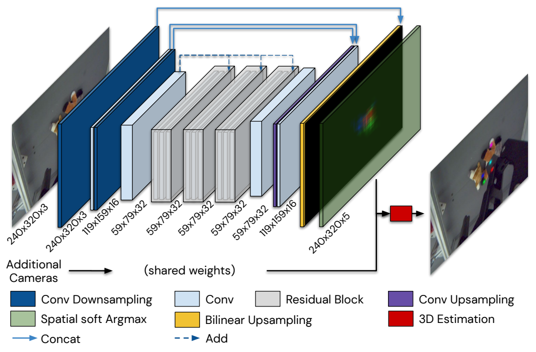

Figure 10: Network architecture used.

The input to the model is a single camera image and the output is a heat-map over all keypoints from that point of view. Since we have multiple cameras providing images at each timestep we apply the model with the same weights to all camera views. The network diagram is presented in Fig. 10.

On the lowest resolution we apply blocks inspired by ResNet[34] with 3 convolutional layers and 32 channels in the intermediate layers. We use a relatively small number of channels (16 to 32), as using more did not increase accuracy, but decreased the framerate below our requirements for control.

Appendix B Detection weight

As mentioned in Section 3.2.1 It is not always desirable to incorporate all predictions from all cameras. Additionally, it is often desirable to put a higher weight on the camera views with the more confident predictions. We explored several explicit ways to predict the confidence from the network, but found that the most effective way was to look at the variance var_{ck} of the prediction heatmap P_{ck} itself. We defined the following confidence measure to weigh the importance of each heatmap:

$$w_{ck} = \mathrm{sigm}(3\tanh(5(1 - \sqrt{var_{ck}}/2/\sigma))) \tag{5}$$

The details of this function are arbitrary, but the important property of this function is that it decreases as the width of the distribution increases above roughly 2σ. Crossing below this value means that the variance of the prediction is approaching the ground truth variance we are regressing to. Also this function never reaches 0 which helps with numerical stability. This weight factor is then used to multiply the quadratic loss associated with this detection in Eq. 2.

Appendix C Hardware details

In all physical experiments we used 4 cameras. They were RGB Basler daA1280-54ucm S-Mount with Evetar Lens M118B029528W F2.8 f2.95mm 1/1.8". The robot was a Sawyer with a 2f-85 Robotiq gripper and FT 300 Force Torque Sensor which was used for a velocity admittance control to ensure the safety of the system. We used custom 3D printed fingers to grasp the cable in a stable way.

Appendix D Simulated environment details

All of our experiments used pixel error as the main metric. We would like to provide some further information to allow a more intuitive understanding. In Section 4.1 1 pixel corresponded to about 2.8mm in the middle of the workspace. Therefore 30px are about 8.4cm. The size of the box was 19.9x1.7x10.5cm.

The provided measurements above apply to the experiments in Section 4.2 as well. 10px corresponds to 2.8cm which is relatively small compared to the overall size of the box. The shoes in section Section 4.3 have realistic sizes, *i.e.* about 25-30cm long. Images of these shoes can be seen in Fig. 11.

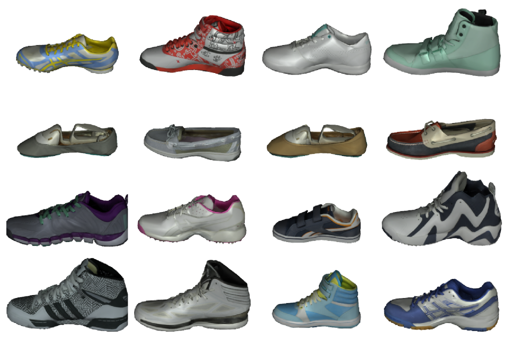

Figure 11: Shoes used in the generalization experiments. Top 3 rows were used in training and bottom row was used for evaluation.

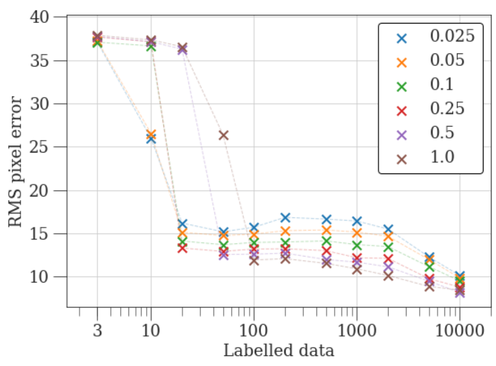

Figure 12: Choosing unsupervised loss weight α.

Appendix E Choosing weight for self-supervised loss

In Section 4.1 we saw instabilities for small amounts of unsupervised data. Therefore we explored how these effects change with the relative weight α between the supervised and unsupervised losses. We used the domain shift dataset from Section 4.2 in the *full* mode with 10k unsupervised data points.

In Fig. 12 we see that the training can exhibit 2 main behaviours. If we don't provide enough labelled data the model collapses to a trivial solution and is unable to use the unlabelled data well. On the other hand with enough labelled data the model performs well with a wide range of values of α, but works best with $\alpha = 1$, *i.e.* large weight on the unsupervised dataset. This however is not true for the small amounts of labelled data where decreasing the weight of the self-supervised loss helps to stabilize the training. Lowering α prevents the model from collapsing on simple 3D consistent solutions such as tracking the centre of the box which would be preferred by the self-supervised loss in the absence of labels. Overall as we decrease the amount of unlabelled data we see that the optimal unsupervised loss gradually decreases. In other experiments we used a weight of $\alpha = 0.5$.