Perception 1

Tuesday, 16 March 2021 18:17

① Transporter Networks (Transporter_google):

A simple model architecture that rearranges deep features to infer spatial displacements from visual input - which can parametrize robot actions.

+: makes no assumptions of objectness (e.g. pose, model or keypoints).

+: they rely only on information contained within partial RGB-D data from demonstrations.

-: works for pick-and-place, and other tasks that are parametrized by two end-effector poses.

Transporter networks use Fully Convolutional Networks (FCN) to model an action-value function Q.

② Unsupervised learning of object keypoints for perception and control (Transporter_deepmind):

Transporter: a neural network architecture for discovering concise geometric object representations in terms of keypoints or image-space coordinates.

+: learns from videos in a fully unsupervised manner; by transporting learnt image features between video frames using a keypoint bottleneck.

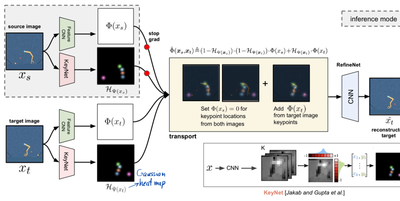

Figure 1: **Transporter.** Our model leverages object motion to discover keypoints by learning to transform a source video frame (x_s) into another target frame (x_t) by *transporting* image features at the discovered object locations. During training, spatial feature maps $\Phi(x)$ and keypoints co-ordinates $\Psi(x)$ are predicted for both the frames using a ConvNet and the fully-differentiable *KeyNet* [17] respectively. The keypoint co-ordinates are transformed into Gaussian heatmaps (same spatial dimensions as feature maps) $\mathcal{H}_{\Psi(\boldsymbol{x})}$. We perform two operations in the transport phase: (1) the features of the source frame are set to zero at both locations $\mathcal{H}_{\Psi(\boldsymbol{x}_s)}$ and $\mathcal{H}_{\Psi(\boldsymbol{x}_t)}$; (2) the features in the source image $\Phi(x_s)$ at the target positions $\Psi(x_t)$ are replaced with the features from the target image $\mathcal{H}_{\Psi(\boldsymbol{x}_t)} \cdot \Phi(x_t)$. The final refinement ConvNet (which maps from the transported feature map to an image) then has two tasks: (i) to inpaint the missing features at the source position; and (ii) to clean up the image around the target positions. During inference, keypoints can be extracted for a *single* frame via a feed-forward pass through the *KeyNet* (Ψ).

+: Transporter keypoints provide a concise visual state-abstraction which enables faster learning → Data-efficient RL.

+: Can be used to learn new action space kX4 intrinsic reward functions using the keypoint locations Training → reduce the search space using a fixed controllability policy π_{Qgap} to select the single most controllable keypoint, (picks keypoints for which actions lead to more change).

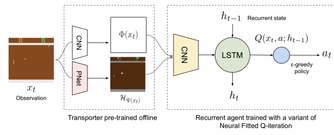

Figure 5: **Agent architecture for data-efficient reinforcement learning.** *Transporter* is trained off-line with data collected using a random policy. A recurrent variant of the neural-fitted Q-learning algorithm [30] rapidly learns control policies using keypoint co-ordinates and features at the corresponding locations given game rewards.

-: doesn't handle moving backgrounds.

③ Supervised training of Dense Object Nets (sup-DON):

-: uses the 3D model of the object.

Rely on Laplacian Eigenmaps (LE) to embed the 3D model of an object into an optimally generated space.

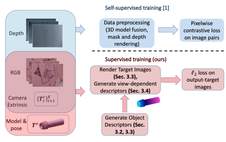

Figure 4: Comparison of the self-supervised and the supervised training (ours) pipelines. The first utilizes depth images to find pixel correspondences and relies on the contrastive pixel-wise loss. Ours exploits the object model and its pose in world coordinates to render training target images, which will be used in the loss function (Eq. (8)).

④ S3k: Self-Supervised Semantic keypoints (S3k):

The goal: learn an explicit 3D representation that captures task-relevent object geometry, without requring depth-sensing or full supervision.

+: Can be trained up to 1-5 millimeter accuracy.

+: Uses multiple camera views.

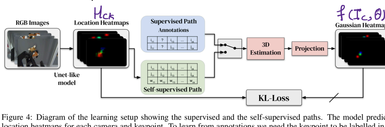

Figure 4: Diagram of the learning setup showing the supervised and the self-supervised paths. The model predicts location heatmaps for each camera and keypoint. To learn from annotations we need the keypoint to be labelled in at least 2 out of 4 cameras. Otherwise we use the coordinates and uncertainties 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.

-: Needs camera calibration to compute the 3D location of all keypoints for supervised keypoint training.

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

all camera images; all keypoints

Self-supervised keypoint training:

Estimated keypoints loc. $\leftarrow \tilde{x}_k = \left(\sum_{c}^{C} I - \hat{d}_{ck}\hat{d}_{ck}^T\right)^{-1} \left(\sum_{c}^{C} v_{ck}\left(I - \hat{d}_{ck}\hat{d}_{ck}^T\right) a_c\right)$

the normalized direction of the ray from camera c. through keypoint; the location of camera

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

$\tilde{H}_{ck}$ created from $\tilde{x}_k$

Total loss: $\mathcal{L}_{total} = \mathcal{L}_{sup} + \alpha \mathcal{L}_{unsup} + \lambda \|\theta_{non\,bias}\|^2$

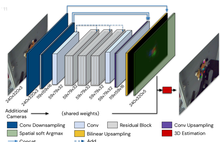

Figure 5: Network architecture used.

+: 10 labelled samples (40 images) is sufficient to obtain a good model if given enough unlabelled data.