CIS 580, Machine Perception, Spring 2019
Homework 6
Due: Friday May 3rd 2019, 5pm

Instructions

- This is an individual homework and worth 100 points
- You must submit your report on Gradescope, the entry code is 96JGNN. We recommend that you use LaTeX, but we will accept scanned solutions as well.
- You must submit code through turnin on a CETS run server (there is no autograder).
- Start early! If you get stuck, please post your questions on Piazza or come to office hours!
- The following files must be submitted via turnin: `cis580_2019_hw6.ipynb`.

Homework

1 Step 1: Keypoint Localization

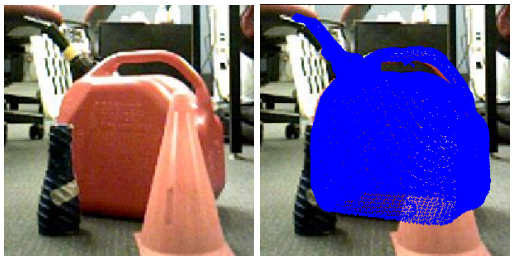
In the first step, you will train a heatmap-based neural network that given an image of an object, it estimates the location of the keypoints in the image. For this part, we provide the keypoint annotations for each image, so that you can train the network in a fully supervised manner. This means that you will only need to fill specific parts of the training code and follow the training procedure.

For the keypoint representation, we will use heatmaps. Given an image of input size 256×256, the output heatmaps for k keypoints will be a 3D tensor of size $k \times 64 \times 64$. This is effectively a collection of k 2D heatmaps with dimensions 64×64. Each 2D heatmap corresponds to one keypoint and is responsible to localize this particular keypoint in the image.

During training, we synthesize the heatmaps by identifying the location of each keypoint on the 2D image and placing a 2D Gaussian centered on this location on the corresponding heatmap. The Gaussian models our confidence for the location of the keypoint. In other words points that are close to the actual location have high likelihood, while points that are far from the keypoint location have small likelihood.

Having synthesized these heatmaps, we have created input-output pairs for training. We use RMSprop for the optimization, the learning rate is equal to $2.5e-4$, while we set the batch size equal to 8. To avoid overfitting, we also use data augmentation, in the form of scale augmentations and 2D rotations in the input.

Your job is to write the training loop for the network training. We handle all the data preprocessing and ground truth generation for you. There are two parts inside the function `train()` that you need to implement, which are clearly indicated in the provided code. For a demonstration of a simple training loop in PyTorch, you can take a look here. For the report you need to demonstrate some successful keypoint localizations on the test images.



(a) (b)

Figure 1: Left: Input image. Right: Reconstructed mesh overlayed on the RGB image

2 Step 2: Pose Optimization

For the second step, we use the coordinates of detected keypoints to estimate the 6DoF pose of the object. To do this, we need to align the CAD model of the object to the detected 2D keypoints, i.e., estimate the rotation $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and the translation $\boldsymbol{T} \in \mathbb{R}^{3\times1}$ between the object and the camera frame such that the projection of the CAD model aligns with the object in the image. Let us denote with $\boldsymbol{P} \in \mathbb{R}^{3\times k}$ the $k = 10$ keypoints on our object and with $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ the camera parameters. If we detect 2D keypoints $\boldsymbol{W} \in \mathbb{R}^{2\times k}$ with localization confidence d_i for keypoint i, we can also define the diagonal matrix:

$$\boldsymbol{D} = \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_p \end{bmatrix}. \tag{1}$$

Given these definitions, we can express our goal as the optimization problem:

$$\min_{\boldsymbol{R},\boldsymbol{T}} \left\| \xi(\boldsymbol{R},\boldsymbol{T})\boldsymbol{D}^{\frac{1}{2}} \right\|_F^2, \tag{2}$$

where $\xi(\boldsymbol{R},\boldsymbol{T})$ are the fitting residuals between the detected 2D keypoints and the projection of the corresponding 3D keypoints:

$$\xi(\boldsymbol{R},\boldsymbol{T}) = \boldsymbol{W} - \boldsymbol{\Pi}(\boldsymbol{K}(\boldsymbol{R}\boldsymbol{P} + \boldsymbol{T})). \tag{3}$$

The operator $\boldsymbol{\Pi}$ is used to indicate the perspective projection:

$$\boldsymbol{\Pi}([X, Y, Z]) = [X/Z, Y/Z]. \tag{4}$$

You can easily solve this simple optimization problem with gradient descent on PyTorch, and recover the rotation $\boldsymbol{R}$ and translation $\boldsymbol{T}$ parameters.

You need to implement:

1. the function `heatmaps_to_locs()` that takes the heatmaps from the network output and estimates the locations of the maximum values along with the magnitude of the maxima.

2. the function `pose_optimization()` that estimates the rotation $\boldsymbol{R}$ and translation $\boldsymbol{T}$ given the network's detections. To avoid enforcing constraints for the validity of the rotation matrix, we suggest optimizing over the rotations in the axis-angle representation (using the `Rodrigues()` function we have provided).

For the report, you need to present some (e.g., 5) successful reconstructions using the mesh visualization on the test images (as in Figure 1).