Pose estimation using a variety of techniques

Keegan Go
Stanford University
keegango@stanford.edu

Abstract

Vision is an integral part robotic systems – a component that is needed for robots to interact robustly and precisely with their environment. For this project, I am interesed in using vision to obtain pose (position and orientation) estimates for objects. The goal is to develop a simple vision system that can be attached to our robots which will give us realtime estimates of positions for specific objects. I implemented pose estimation using convnets on a very simple dataset and compare this against a traditional multiview geometry approach.

1. Introduction

In manipulation tasks, such as picking up or moving objects, it is helpful to obtain estimates of the position and orientation of those involved objects. At the Stanford robotics lab, we have previously done a lot of work on control strategies that use force feedback only. However, many of these strategies are not robust in the sense that they require that the initial alignment error is within a reasonable range. In a controlled setting, this works great, but it does not generalize well when we want the robot to move around freely. One solution, is to use vision methods to detect desired objects, and estimate their positions so that we can still perform an alignment prior to interacting with them.

Such a system has a number of requirements. It should be relatively cheap to train in both a time and effort, since the set of desired objects may change over time. At estimation time, it should also run quickly with a relatively low computational power since we'd like to deploy the system on mobile platforms.

This project is divided into two parts. The first part attempts to do pose estimates on some simple generated image data using a convnet approach. The second uses a multiview 3d reconstruction approach. Details for each method are included in their respective session, and a third section discusses the differences between the two from an overall implementation standpoint. (Note: the division between the first two sections separates the work done for cs231n/a, respectively).

2. Pose estimation with convnets

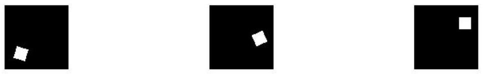
2.1. Background

A search for ”pose estimation” within the domain of convnets brings up a large number of papers that deal with recognizing the pose of humans. In [3], the authors localize the position of different segments of the human body, by individually classifying and localizing body parts using a sliding window and then linking these parts together in a way consistent with the structure of the human body. In [4], a regression model is used to localize the body parts. To improve results, multiple frames are used together.

Neither of these works describe what we are looking for in our version of pose estimation, however, they do suggest possible methods to use. In our problem, we will assume that objects are rigid and represented by a single part, rather than being dynamic like the human body. While this simplifies the problem in the respect that we do not need to resolve the structure of multiple parts, the problem we focus on is different since we would ideally like to obtain 3d information from the image. But using the same ideas, there are two approaches to consider. The first is to divide an object to be detected into ”parts”, localize each part and then use geometry to compute a rotation and position consistent with the location of the detected parts. Note that in this case, the convnet would essentially be used as a feature detector. A more appealing option is to use regression to estimate the parameters directly. This avoid the hassle adding additional information about the location of different parts to the images of the object we are working with. Thus, a dataset may be obtained by fixing the object in some location, and taking pictures of it at known relative locations.

2.2. Approach

I generated a simple dataset. To simplfy the problem, I considered only images of size 80×80 of a black 2d plane with a single white square placed on it. The position of the square was generated from a uniform random distribution, with bounds such that the square is always entirely inside the image. The square is then rotated an angle drawn uniformly $[0°, 90°]$ to avoid the problem of rotational symmetry. Examples of the generated images are shown below. I trained on a set of 1000 such images, and tested on 200 images. The number of training and test images were kept small to mirror a reasonable amount that might be collected by hand on a real 3d object if this method were to be applied.



I then formulated the problem using regression, that is, for the i-th example, the convnet will predict $q_i = (\theta_i, x_i, y_i)$, the angle by which the square is rotated, as well as an offset along the x, y-axes. The loss L of a prediction of N examples is defined using an l_2-norm penalty

$$L = \frac{1}{N} \sum_{i=1}^{N} \|q_i - q_i^{\text{true}}\|^2$$

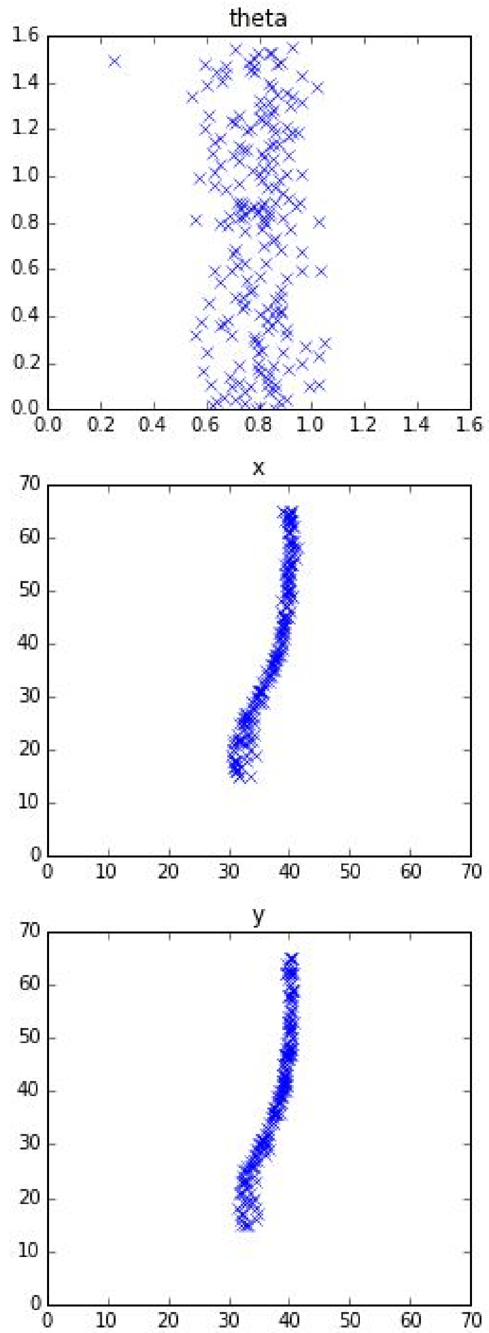
where q_i^{true} is the true pose of the object. The gradient is readily computed.

2.3. Results

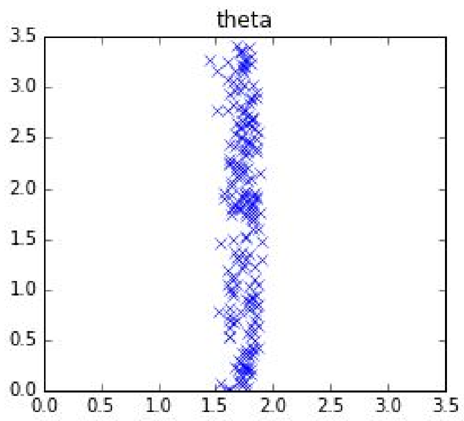
Note that the simple baselines of least squares does not work here, since with there are more pixels than we have training examples. The table below summarizes the mean-squared-errors for constant predictors giving a baseline to compare our loss during training against.

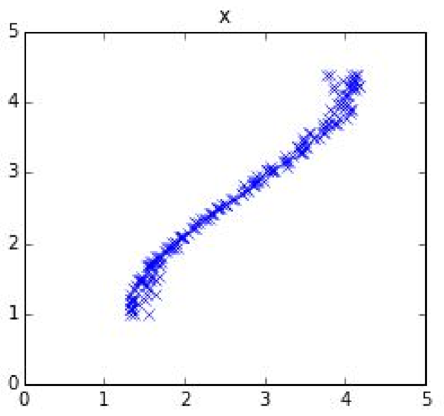
Prediction	Mean-square-error
θ	0.2078
x	221.0
y	193.3

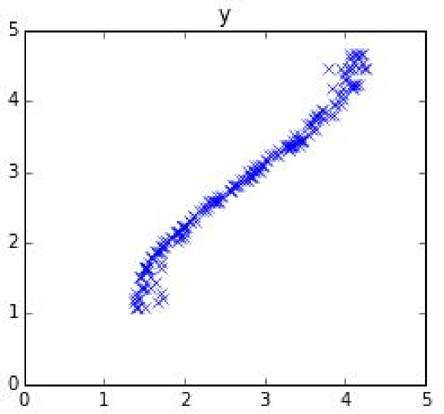
I began with a simple two layer network (conv-relu-pool, affine). In this case, the training loss and test loss both convered to around 200, half that expected using constant predictors (i.e. not very good). A useful plot is to look at the predictions for each parameters versus the truth value (the equivalent of a confusion matrix for regression). The plot is given below.



As the figure shows, none of the parameters are predicted very well. Each of them is relatively, vertical indicating that the convnet is defaulting to a nearly constant predictor. One of the issues is the difference in scaling between the different parameters we are trying to predict. To fix this, I normalized the parameters so that they had approximately the same variance, and then retrained the network. After normalizing the data I reran the network and obtained the following plot of prediction versus true value.







Interestingly, while normalizing the data greatly improved prediction of the position, it did not help prediction of the orientation. I trained using a deeper (4 layer) network but the results remained roughly the same. [3] suggests that the pooling layers may reduce the spatial precision which could also hamper the ability for the network to predict orientation. There may also be ambiguity in the data due to rotational symmetry of the sqaure.

3. Pose estimation using multiview geometry

3.1. Background

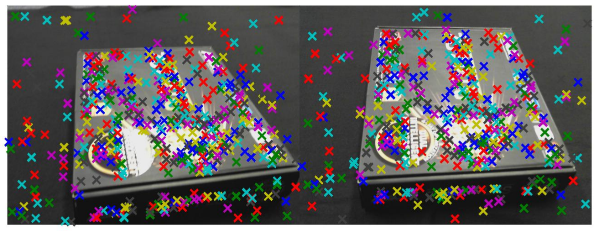
The 3d pose estimation for general objects is a well studied problem in featured based vision methods. A well documented process is given by [2] and the higher level view is given by [1]. The method described makes use of a number of techniques which will be fully described in the approach section, but essentially the ”training” part of the method consists of two parts. The first is to find matching points across multiple images. Second, these matches are used to impose constraints on the images during reconstruction. The process, and the subsequent pose identification has been shown to work in realtime.

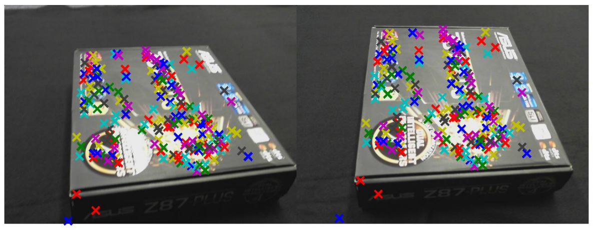
3.2. Approach/Results

The first step is obtain images of the object. [2] suggests using 20 images, often 10 are sufficient. Example images that I used are shown below.

I then extraced SIFT features from each image. For each pair of images, we can then count the number of matches that are sufficiently close between images. For a typical pair of image, we typically obtain a large number of matches some of which are false matches. Therefore the next is to ”sift” out the good matches, by using the epipolar constraint.

The method applied is to select a random subset of matching points, and perform the 8-point algorithm to obtain a candidate $\hat{F}$ for the fundamental matrix F. If the matches are all good, then we expect that $\hat{F}$ is close to F and thus matching points in one image will be close to the epipolar lines defined by F from the points in the paired image. Repeating this method and taking the best $\hat{F}$ as deteremined by the smallest epipolar line constraint violations gives a set of filtered points that are likely to be matches. An example of this matching is shown below, where the top image is before filtering and shows a number of extraneous matches, and the second image has most of these incorrect matches removed.

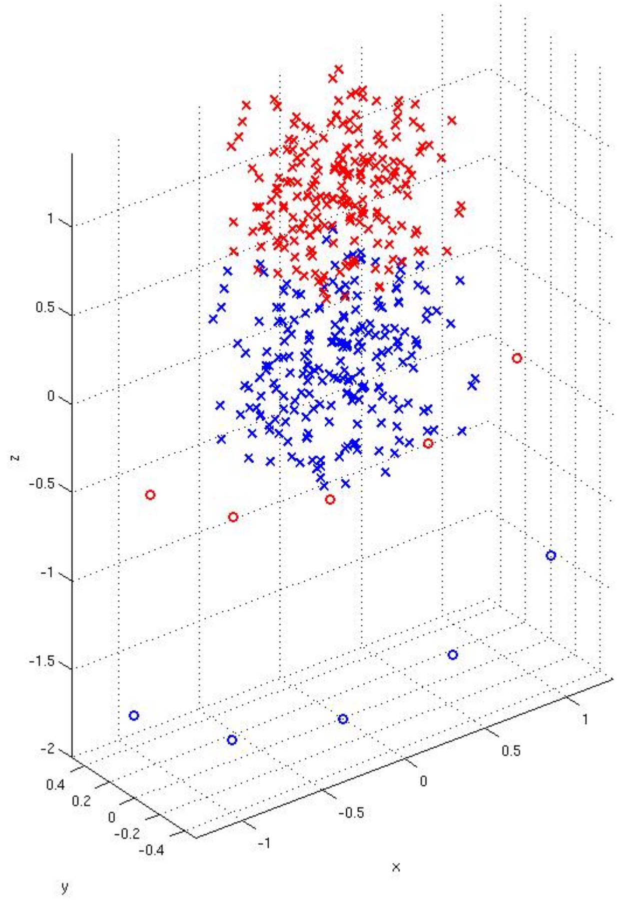




One matches between pairs of images have been extracted, we can find matches between multiple view by transivitity. At this point we are ready for reconstruction.

The process of reconstruction works by choosing a number of points in 3d such that the projection of each 3d point onto each image is consistent with the matches found. I used Levenberg-Marquardt iteration to find a solution to this nonlinear system. An example of the solution is shown to the right.

The blue xs are the original points and the blue circles are the positions of the cameras. The red xs are the reconstructed points. The two have very similar shapes up to a scaling factor. This scaling factor can then be accounted for by fixing the distances between reconstructed points, but the difference makes the reconstruction more convenient for viewing.



4. Conclusion

Of the two methods described above, the multiview approach seems more suited to this version of pose estimation. It requires less data to be collected and labelled making it easier to use different objects.

However, the convnet approach remains an attractive prospect. While I was not able to get the setup to work effectively, the possibility of learning a higher level representation of 3d rotation and position is exciting since such a convent would be (hopefully) scalable to many different objects using transfer learning.

References

[1] Gordon and Lowe. What and where: 3d object recognition with accurate pose, 2006.

[2] I. Gordon. Augmenting reality, naturally, 2000.

[3] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler. Learning human pose estimation features with convolutional networks. *arXiv preprint arXiv:1312.7302*, 2014.

[4] Pfister, Simonyan, Charles, and Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos, 2014.