Pose Estimation Based on 3D Models

Chuiwen Ma, Liang Shi

1 Introduction

This project aims to estimate the pose of an object in the image. Pose estimation problem is known to be an open problem and also a crucial problem in computer vision field. Many real-world tasks depend heavily on or can be improved by a good pose estimation. For example, by knowing the exact pose of an object, robots will know where to sit on, how to grasp, or avoid collision when walking around. Besides, pose estimation is also applicable to automatic driving. With good pose estimation of cars, automatic driving system will know how to manipulate itself accordingly. Moreover, pose estimation can also benefit image searching, 3D reconstruction and has a large potential impact on many other fields.

Previously, most pose estimation works were implemented by training on manually labeled dataset. However, to create such a dataset is extremely time-consuming, laborsome, and also error-prone because the labelization might be subjective. Therefore, the training datasets in existing works are either too small or too vague for training an effective classifier. In this project, we instead utilized the power of 3D shape models. To be specific, we built a large, balanced and precisely labeled training dataset from ShapeNet [6], a large 3D model pool which contains millions of 3D shape models in thousands of object categories. By rendering 3D models into 2D images from different viewpoints, we can easily control the size, the pose distribution, and the precision of the dataset. A learning model trained on this dataset will help us better solve the pose estimation task.

In this work, we built a pose estimation system of chairs based on a rendered image training set, which predicts the pose of the chair in a real image. Our pose estimation system takes a properly cropped image as input, and outputs a probability vector on pose space. Given a test image, we first divide it into a $N \times N$ patch grid. For each patch, a multi-class classifier is trained to estimate the probability of this patch to be pose v. Then, scores from all patches are combined to generate a probability vector for the whole image.

Although we built a larger and more precise training dataset from rendered images, there is an obvious drawback of this approach — the statistical property of the training set and the test set are different. For instance, in the real world, there exists a prior probability distribution of poses, which might be non-uniform. Furthermore, even for features from the same pose, real image features might be more diverse than rendered image features. In this paper, we proposed a method to revise the influence of the difference in prior probability distribution. Detailed methods and experiment results are shown in the following sections.

2 Dataset, Features and Preprocessing

2.1 Training Data

As we mentioned in Section 1, we collected our training data from ShapeNet, a 3D shape model database, which contains 5057 chair models. For each model, we rendered it on 16 viewpoints, evenly distributed on the horizontal circle, shown in Figure 1.

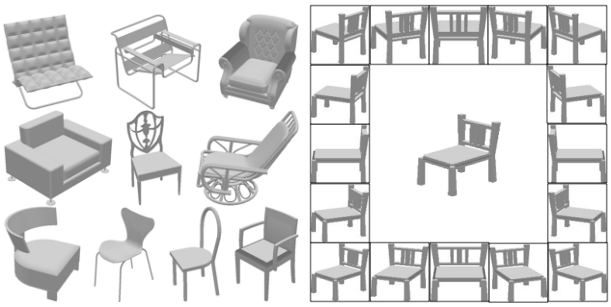

Figure 1: Chair models and rendering process

We chose 4000 models, accordingly 64,000 images to build the training dataset, and leave the rest 1057 models to be our rendered image test set. When extracting image features, we first resize the images to 112×112 pixels, and then divide it into 6×6 overlapped patch grid, with patch size 32×32 and patch stride 16 on both axes. After that, we extract a 576 dimensional HoG features [3] for each patch, so the whole image can be represented by a 20736 dimensional feature vector. Those 64,000 feature vectors constituted our training dataset.

2.2 Test Data

To better evaluate the performance of our learning algorithm, we built three different test sets with increasing test difficulty. They are rendered image test set, clean background real image test set and cluttered background real image test set.

Rendered image test set consists of 1057×16 rendered images, which also comes from ShapeNet. Clean background and cluttered background real image test sets are collected from ImageNet [4], containing 1309 and 1000 images respectively, both with manually labeled pose ground truth. Some sample images are shown in Figure 2. Obviously, these three datasets are increasingly noisy and difficult to tackle.

Figure 2: Clean background & cluttered background

For the test sets, we used the same scheme to process the image as the training set. That is, convert each image into a 20736-dimension HoG feature.

3 Model

Rather than using global image feature as the input of classification, our pose estimation model is patch-based. By dividing image into patches and training a classifier for each patch, our model can be more robust to occlusion and background noise. Also, this approach reduced the feature dimension, thus reduced the sample complexity for each classifier. Actually, we did try the global method, while the classification accuracy is 20% lower than patch based method. The mathematical representation of our patch based model is as follows.

Define F_i as the HoG feature of patch i, $I = (F_1, \cdots, F_{N^2})$ to be the HoG feature of the whole image, $\mathcal{V} = \{1, \cdots, V\}$ to be the pose space. For each patch, we build a classifier, which gives a prediction of the conditional probability $P(v|F_i)$. To respresent $P(v|I)$ in $P(v|F_i)$, $i = 1, \cdots, N^2$, we assume $P(v|I) \propto \prod_{i=1}^{N^2} P(v|F_i)$. So, we can calculate $P(v|I)$ and the according $\bar{v}$ using the following formula:

$$P(v|I) = \frac{\prod_{i=1}^{N^2} P(v|F_i)}{\sum_{v=1}^{V} \prod_{i=1}^{N^2} P(v|F_i)}$$

$$\bar{v} = \arg\max_v P(v|I)$$

In sum, our model takes F_i, $i = 1, \cdots, N^2$ as input, and outputs $P(v|I)$ and $\bar{v}$.

4 Methods

4.1 Learning Algorithms

4.1.1 Random Forest

In this project, we choose random forest [1] as a primary classification algorithm based on following considerations:

- Suitable for multiclass classification.
- Non-parametric, easy to tune.
- Fast, easy to parallel.
- Robust, due to randomness.

During classification, 36 random forest classifiers are trained for 36 patches. As a trade off between spatio-temporal complexity and performance, we set the forest size to be 100 trees. We also tuned the maximum depth of trees using cross-validation, where the optimal depth is 20. When classification, each random forest outputs a probability vector $P(v|F_i)$. After Laplace smoothing, we calculated $P(v|I)$, estimated the pose to be $\bar{v} = \arg\max_v P(v|I)$.

4.1.2 Multiclass SVM

In binary classification, SVM constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space that will separate data with different labels as wide as possible. In C-SVM model, the hyperplane is generated by maximizing the following function

$$\min_{\gamma,w,b} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \xi_i$$

$$s.t. \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i,$$
$$\xi_i \geq 0, \ i = 1 \ldots, m.$$

two important factors will shape the outcome of model — kernel and soft margin parameter C. Kernel defines a mapping that projects the input attributes to higher dimension features, which could often convert the non-separable data into separable. C determines the trade-off between the training error and VC dimension of the model, the smaller C is, the less effect will the outliers exert on the classifier.

In our problem, we face a multiple classification problem which can not be addressed by building a single SVM model. Here, we apply two methods to solve it: 1. One-versus-Rest (OvR) 2. One-versus-One (OvO). Both these two methods reduce the single multiclass problem into multiple binary classification problems. OvR method train classifiers by separating data into one exact label and the rest, which results in n models for n-label data, then it predicts the result as the highest output. In OvO approach, classification is done by a max-wins voting strategy. For n-label data, $n(n-1)/2$ classifiers are trained, with each trained by picking 2-label data from entire n-label data. In prediction, each classifier assigns the input to one of the two labels it trained with, and finally the label with the most votes determines the instance classification.

4.2 Optimization

Constructing training dataset from rendered images has many advantages, but there are also drawbacks. As I mentioned in Section 1, the prior probability of pose in real images can be highly different from that in rendered images. As we know, pose distribution in the training set is uniform, however, in real images, there are far more front view chairs than back view. Fortunately, this difference can be analyzed and modeled as follows.

4.2.1 Probability Calibration

In classification step, each classifier C_i will output a probability vector $\tilde{P}(v|F_i)$. Using Bayesian formula, we have:

$$\tilde{P}(v|F_i) = \frac{\tilde{P}(v)\tilde{P}(F_i|v)}{\tilde{P}(F_i)}$$

Here, $\tilde{P}(v)$, $\tilde{P}(F|v)$ and $\tilde{P}(F)$ are learned from training data. Whereas, the real $P(v|F_i)$, which satisfies the following formula, could be different from $\tilde{P}(v|F_i)$. Here, $P(v)$, $P(F_i|v)$ and $P(F_i)$ are distributions in the test set.

$$P(v|F_i) = \frac{P(v)P(F_i|v)}{P(F_i)}$$

Assume the training data and the test data have at least some similarity. Specifically speaking, assume $P(F_i|v) = \tilde{P}(F_i|v)$, $P(F_i) = \tilde{P}(F_i)$, then we have:

$$P(v|F_i) = \tilde{P}(v|F_i)\frac{P(v)}{\tilde{P}(v)} \propto \tilde{P}(v|F_i)P(v)$$

To recover $P(v|F_i)$, we just need to achieve a good estimation of $P(v)$. One possible method might be randomly choosing some samples from the test set, and manually label the ground truth of pose, regard the ground truth pose distribution of samples as an estimation of global $P(v)$. However, we still need to do some "labor work".

Noticing the above formula can also be written as:

$$\frac{P(v|F_i)}{P(v)} = \frac{\tilde{P}(v|F_i)}{\tilde{P}(v)}; \ \tilde{P}(v) = \frac{1}{V}, \forall v \in \mathcal{V}$$

we came up with another idea to automatically improve the classification result. For $\tilde{P}(v|F_i)$, we have:

$$P(v) > \frac{1}{V} \Rightarrow \tilde{P}(v|F_i) < P(v|F_i)$$
$$P(v) < \frac{1}{V} \Rightarrow \tilde{P}(v|F_i) > P(v|F_i)$$

That means, when testing, frequently appeared poses are underestimated, while uncommon poses are overestimated. Here, we will propose an iterative method to counterbalance this effect. Basically, we will use $\tilde{P}(v|F_i)$ to generate an estimation $\tilde{P}(v)$ of the prior distribution; assume $P(v)$ and $\tilde{P}(v)$ have similar common views and uncommon views (in other words, $P(v)$ and $\tilde{P}(v)$ have the same trend); smooth $\tilde{P}(v)$ to keep the trend while reduce fluctuation range; multiply the original $\tilde{P}(v|F_i)$ by smoothed $\tilde{P}(v)$; and iteratively repeat the above steps. Finally, due to the damping effect in combination step, $\tilde{P}(v)$ will converge, and $\tilde{P}(v|F_i)$ gets closer to $P(v|F_i)$. Formulation of this iterative algorithm is as follows:

1. Calculate $\tilde{P}(v|I^{(j)})$, $j = 1, \cdots, m$.

$$\tilde{P}(v|I^{(j)}) = \frac{\prod_{i=1}^{N^2} \tilde{P}(v|F_i^{(j)})}{\sum_{v=1}^{V} \prod_{i=1}^{N^2} \tilde{P}(v|F_i^{(j)})}$$

2. Accumulate $\tilde{P}(v|I^{(j)})$ on all test samples to calculate $\tilde{P}(v)$.

$$\tilde{P}(v) = \frac{1}{m}\sum_{j=1}^{m}\tilde{P}(v|I^{(j)})$$

3. Smooth $\tilde{P}(v)$ by factor α.

$$\tilde{P}_{\mathrm{s}}(v) = \frac{\tilde{P}(v)+\alpha}{1+16\alpha}$$

4. Estimate $P(v|F_i)$ by letting:

$$\bar{P}(v|F_i) = \tilde{P}(v|F_i)\tilde{P}_{\mathrm{s}}(v)$$

5. Use $\bar{P}(v|F_i)$ to re-calculate $\tilde{P}(v|I^{(j)})$ in step 1, while remain $\tilde{P}(v|F_i)$ in step 4 unchanged, repeat the above steps.

After several iterations, the algorithm will converge, and we'll get a final estimation $\bar{P}(v|F_i)$ of $P(v|F_i)$.

4.2.2 Parameter Automatic Selection

However, different α will lead to far different converging results, as shown in Figure 3. From experiment results in Figure 4 we observed that if α is too small, viewpoint with the highest probability $\tilde{P}(v)$ will soon beat other viewpoints, and $\tilde{P}(v)$ converges to a totally biased distribution. While, if α is too large, smoothing effect is too strong to make any change of $\bar{P}(v|F_i)$. However, there exists an intermediate value of α to maximize the classification accuracy and result in an optimal estimation $\bar{P}(v|F_i)$. In Figure 3 and 4, it is 0.8.

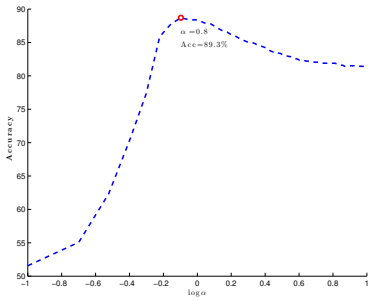

Figure 3: Classification accuracy change w.r.t. α

To solve the optimal α, we conducted deep analysis to the relationship between stable $\tilde{P}(v)$ and α. We found three patterns of relationship between $\tilde{P}(v_j)$ and α, shown in Figure 5. For some viewpoints, $\tilde{P}(v)$

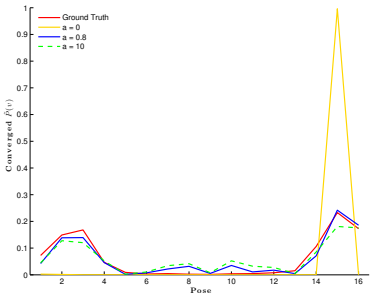

Figure 4: Stable distribution $\tilde{P}(v)$ w.r.t. α

is almost monotonically increasing with respect to α, such as blue curves, some are monotonically decreasing, such as the black curve, while others will decrease after first increase, such as the red curves. Recall the distribution change with α in Figure 4, we found $\tilde{P}(v)$ will first approximate $P(v)$ then be smoothed. So, patterns with turning points are reflection of this trend. Sum on those components, we get Figure 6, and take the turning point of the curve as our estimated $\bar{\alpha}$. Here $\bar{\alpha}$ is 1, very close the optimal value 0.8.

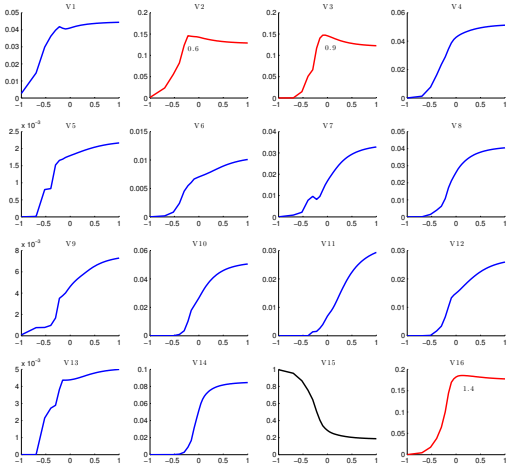

Figure 5: $\tilde{P}(v_j)$ curve with respect to α

5 Results and Discussion

5.1 Classification Performance

Table 1 shows a promising classification results on all three test sets. Under our scheme, OvO SVM

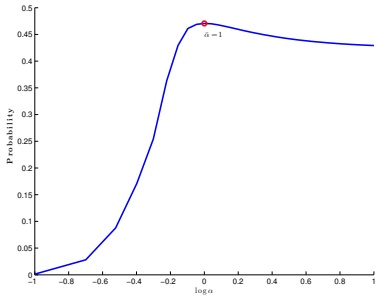

Figure 6: Estimated α

achieves 83% accuracy on clean background real image test set, and 78% on cluttered background test set, which beats other algorithms. After calibrating the conditional probability $\tilde{P}(v|F_i)$ using automatically selected α, performance on clean test set is boosted by 6%, as well 2% on cluttered set. The relatively low improvement on cluttered test set may result from our assumption of $\tilde{P}(F_i|v) = P(F_i|v)$ and $\tilde{P}(F_i) = P(F_i)$ are too strong for cluttered images.

	Render	Clean	Cluttered
RF(%)	**96.16**	80.67	76.80
RF_{opt}(%)	—	88.90	78.70
OvO(%)	95.89	**83.21**	**78.10**
OvO_{opt}(%)	—	**89.12**	**79.94**
OvR(%)	93.42	76.93	70.33
OvR_{opt}(%)	—	83.21	72.98

Table 1: Classification accuracy on three test sets

Figure 7 shows the confusion matrix on three test sets respectively. From left to right, as test difficulty increases, confusion matrix becomes increasingly scattered. On rendered image test set, an interesting phenomenon is that some poses are often misclassified to poses with 90° difference with them, one possible explanation is that the shape of some chairs are like a square. Also, front view and back-view are often misclassified, because they have similar appearance in feature space.

6 Conclusion

In this paper, we proposed a novel pose estimation approach — learn from 3D models. We explained our model in Bayesian framework, and raised a new optimization method to transmit information from test set to training set. The promising experiment

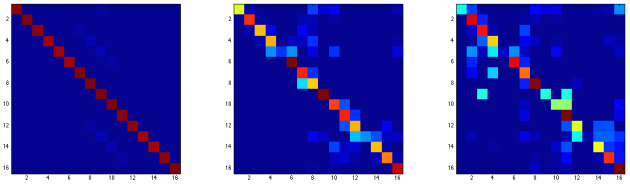

Figure 7: Confusion matrix on rendered, clean, cluttered test sets

results verified the effectiveness of our scheme. More experiment details are omitted due to page limit.

7 Future Work

Our ideas for the future work are described as follows:

- Take into consideration the foreground and background information in the image, fully utilize the information in rendered images.
- Further model the difference between three datasets, revise our inaccurate assumption.
- Learn the discriminativeness of patches, give different weight for different patches.

References

[1] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[2] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: a library for support vector machines." ACM Transactions on Intelligent Systems and Technology (TIST) 2.3 (2011): 27.

[3] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[4] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.

[5] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.

[6] Su, Hao, Qixing Huang and Guibas Leonidas. "Shapenet." <http://shapenet.cs.stanford.edu>