3D Reconstruction of Indoor Scenes using Single Image

Mohammad Haris Baig

An Input Image and it's corresponding depth Map

Goal

The objective of this project is to develop a system, which when presented with a single RGB image of an indoor scene can predict the depth at every pixel of the Image. As was previously stated to accomplish this objective, I intend to try out two kinds of techniques.

1) Non Parametric

2) Parametric

This report discusses the achievements made till date in accomplishing the goals I set out at the beginning of this project.

Dataset

In order to try out the two different approaches I intend to use the following datasets

1) NYU_V2
2) Make3D

Non Parametric Approach

For this approach I have used the NYU_V2 dataset which consists of 1449 RGB Images and their corresponding Depth Maps. I started off by adopting the method suggested by Konrad et.al [1]. According to the approach we compute a set of Features **F**i for every image **i** in our training set.

When given an image, in order to compute its corresponding depth map we compute the set of feature **Ft** for this test image, and compute the Euclidean distance of **F**t from all the images in the training set. Then we choose from the training set **K** neighbors that are closest to the testing image based on the Euclidean distance computed. Then we take the corresponding depth maps for the **K**

neighbors that we selected and for every pixel, consider the median of the depths at that pixel in the set of **K** depth maps that we have. Thus we end up with a depth at every pixel value.

In order to evaluate the performance of this method I used the normalized cross correlation coefficient computed between the ground-truth depth map and the predicted depth map. A higher value of normalized cross correlation coefficient suggests that the only difference between the ground truth and the predicted depth map is a scale factor.

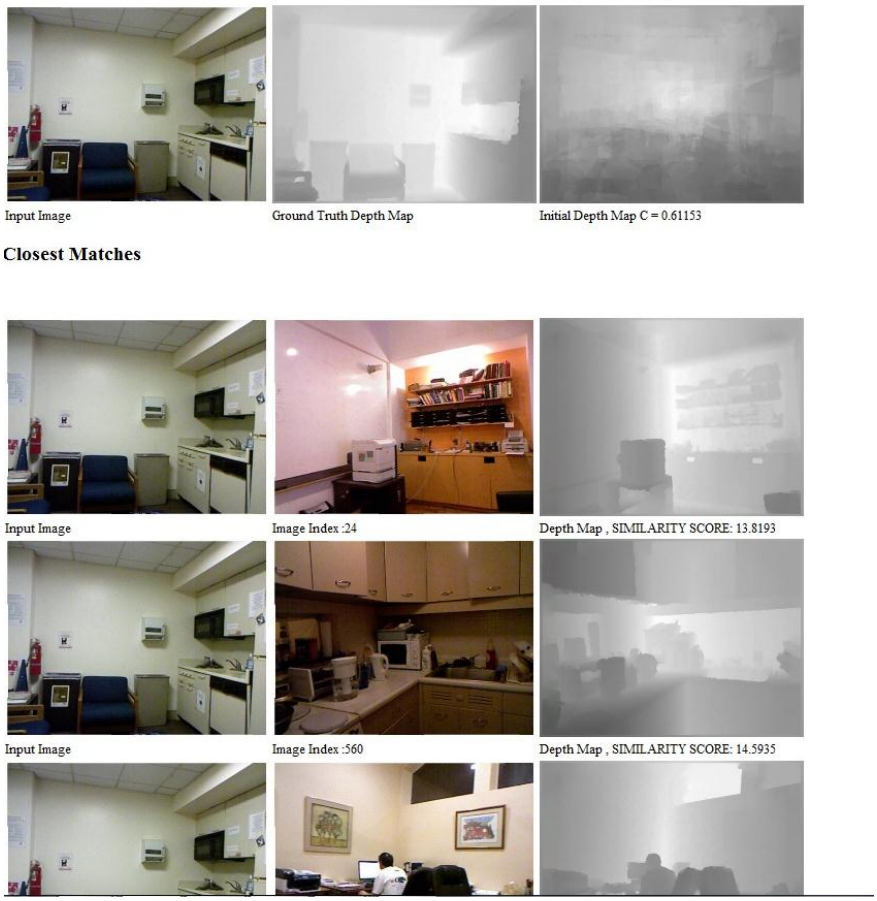

Figure 1. First Row, Left: Input Image, Center: Ground Truth, Right: Predicted Depth. Left: Input Image, Center: K Nearest Neighbors, Right Depth of K Nearest Neighbors

Analysis of Approach

From the above stated method, I concluded that the most important thing about this method was the assumption that similar looking objects have similar depths. What this means for the method is that , if this assumption is to hold true then the method would produce the best results. So in order to validate the approach to the best of my ability, I decided to pursue 3 lines of inquiry.

1) Improving finding of **K** most similar neighbors.
2) Finding out the optimal number of Nearest Neighbors required
3) Finding out the best method, to use the data from the **K** neighbors.

Improving finding of K most similar neighbors

In order to increase the similarity of the matched images, I tried out different features namely

1) HOG (9 Bins)
2) GIST
3) Classemes
4) Pyramid of HOG(PHOG) UNORIENTED (20 bins)
5) Pyramid of HOG(PHOG) UNORIENTED (20 bins)

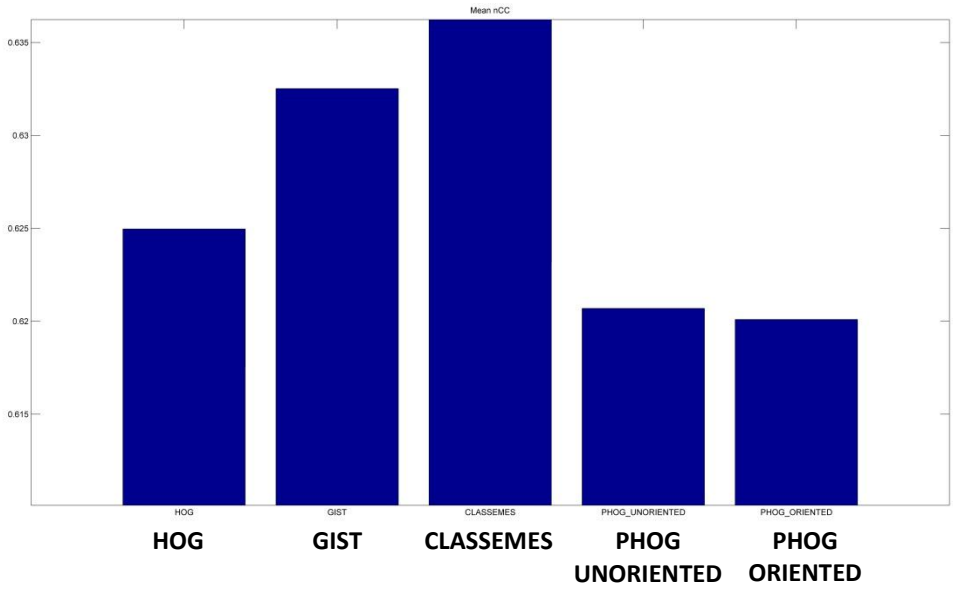

Figure 2. Mean NCC plotted against choice of features used for 30 Nearest Neighbors

To test the goodness of a match I computed 3D structure of the scene using the **K** nearest neighbors suggested by that choice of features. Then based on the value of the normalized cross correlation coefficient I made a claim about which choice of features is better.

Finding out the optimal number of Nearest Neighbors required

This inquiry is more about determining, how big must our **K** be such that we get the best reconstruction. Since this is a non-parametric method, the idea here is that our choice of **K** neighbors should be small enough to prevent under-fitting, and large enough to prevent over-fitting.

For this investigation I evaluated the quality of reconstruction with the use of NCC for the predicted depth map by using different values of **K**.

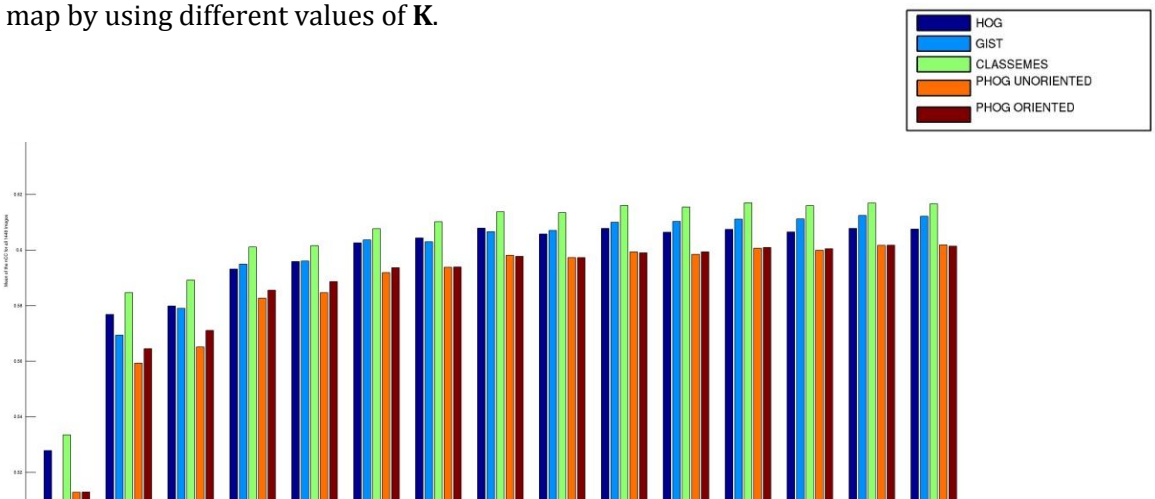

Figure 3. Mean NCC plotted for different feature descriptors for different Values of K with Euclidean Distance

As can be seen beyond a point increasing the value of **K** stopped improving the results of the reconstruction. When NCC flattens out, that is the value of **K** chosen as the best value of **K**.

Finding out the best method, to use the data from the K neighbors

Given that we have chosen the K closest neighbors to the test image, now comes the most important question of what is the optimal method of using the K nearest neighbors suggested.

For this I investigated two kinds of approaches,

1) Global approach (Figure 1)
2) Patch based approach
    1. Same patch
    2. Same Row
    3. All patches

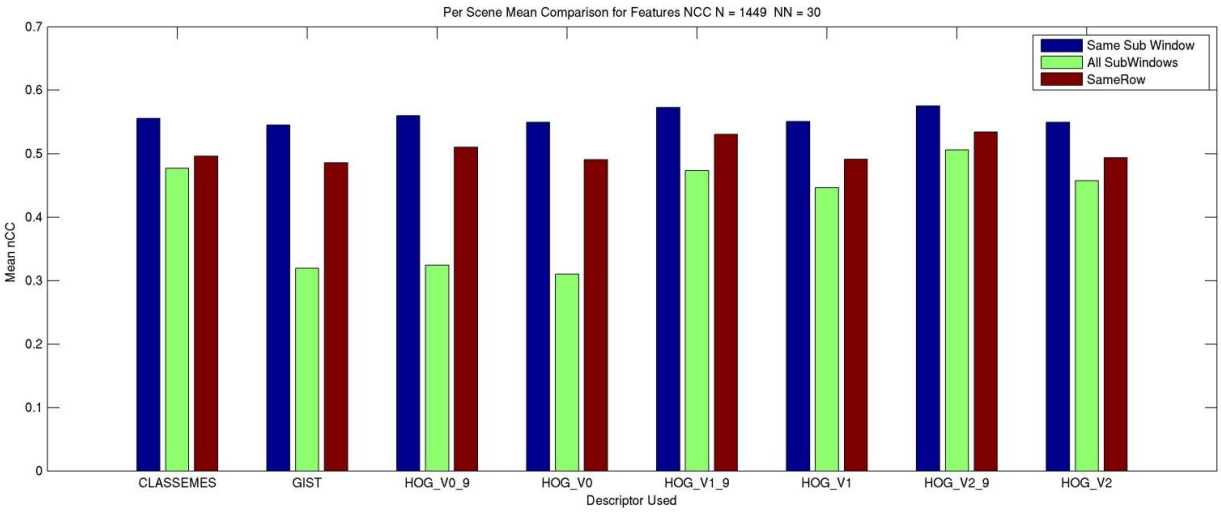

Figure 4. Sowing Mean NCC against different choice of matching strategy for patch based approach for various feature descriptors for K = 30

The motivation behind this experiment was that at a smaller scale we can get a better match then if we try to match the entire image. To conduct the Patch based component of this experiment rather than computing features over the entire image, I computed features for each patch. Now in the first case of the patch based approach I used only the patch at the same location from amongst all the patches from all the training Images. In the second case, for the test patch I used all the patches from the training images in the same row as the test patch and for the last case, I tried matching the test patch against all the patches from the training images.

As the results show, using the same patch yields the best results indicating that this captures some sort of a relationship of similarity in depth at certain locations. Intuitively, this can be thought of as the center part of the image being further and having a wall somewhere in it. The bottom parts being closer and depicting the floor and the top part being closer indicating a ceiling.

Other Experiments Conducted.

Another idea I investigated was to see which distance metric is better for evaluating the distance between features for computing the distance between images or patches. For this experiment I used different distance metrics and computed the depth Map for each test example by computing the K nearest neighbors using different distance metrics. This experiment however was not conclusive as the results suggested that almost all distance metrics performed equally in evaluating the nearest neighbors.

Conclusion of Experiments on Non-parametric Approach

This approach has indicated promising results with average NCC being 0.6 for most experiments with the correct value of K.

What is left to investigate in this area is a better method to utilize the information, whereas [1] suggested the use of median. This only considers the distribution at each pixel and does not enforce any smoothness constraint on either the pixels or smoothness between patches. Also, the choice of median means that the depth predicted must be from the set of depths of the **K** neighbors, an assumption for which we have no justification.

Therefore to find a better way of predicting the depth for a test image that would use smoothness information and did not make any absurd assumptions about the value of depth, we turned to more structured parametric methods.

Parametric Method

One of the key reasons of us using a parametric method was that it provided us with a disciplined way of predicting the depth. With parametric methods we are interested in finding what parameters can be used to describe the relationship between depth and image appearance. To start of, I decided to pursue the method suggested in [2] that involves the use of MRF.

An MRF provides a very natural way of imposing smoothness constraints on adjoining pixels. I first set out to implement the Gaussian MRF as suggested in [2]. For the Gaussian MRF , I started off by using the same dataset that was collected and used by the authors so as to be able to replicate their results and be assured that my code works correctly.

The Gaussian MRF

The Gaussian MRF is made by adding two Gaussian distributions. The first term being representative of predicting the data and the second term being used to enforce smoothness between neighbors based on similarity.

The first step in the implementation was an analytic understanding of the entire Gaussian model that involved finding out how the training parameters were to be obtained, and how the inference was to be done.

$$P_G(d|X;\theta,\sigma) = \frac{1}{Z_G}\exp\left(-\sum_{i=1}^{M}\frac{(d_i(1)-x_i^T\theta_r)^2}{2\sigma_{1r}^2} - \sum_{s=1}^{3}\sum_{i=1}^{M}\sum_{j\in N_s(i)}\frac{(d_i(s)-d_j(s))^2}{2\sigma_{2rs}^2}\right)$$

Data Term Smoothness Term

The next step involved generating the features described in the paper. Whereas for this step I took help from the code provided by the authors at [3], the only use that could be made of this code given the multiple models that it had been modified to adapt to was to use the filter banks that were used in the generation of the features. Based on those filter banks and the description of the features I generated the corresponding features for all the images in the training set.

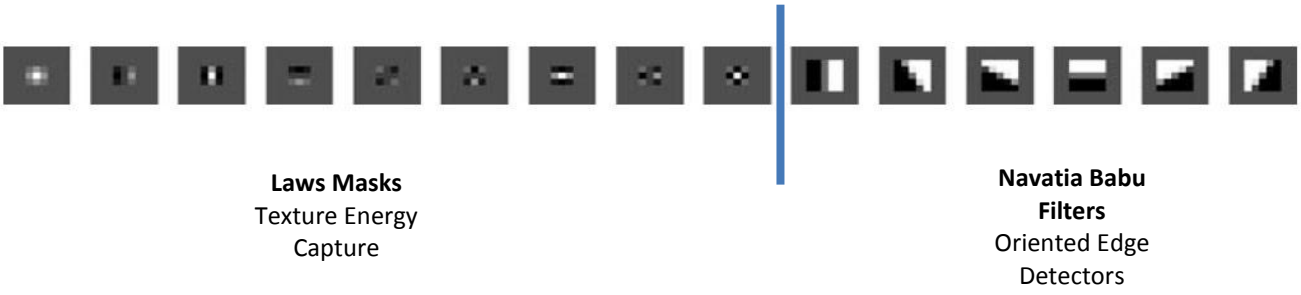

Figure 5. Filter Banks Used

This approach required the generation of two kinds of features, namely absolute features and relative features. Absolute features are used to capture the relationship between depth values and the appearance of the image, whereas relative features are used to capture the smoothness relationship between the image pixels.

Feature Generation

Absolute Features are constructed by applying the filter banks on a patch in the RGB image corresponding to the depth pixel in the depth map of the training set. The filter bank outputs are summed up for all the results on the patch and then concatenated with the neighboring features. This is repeated after downsizing the image multiple times to capture the relationship at multiple scales.

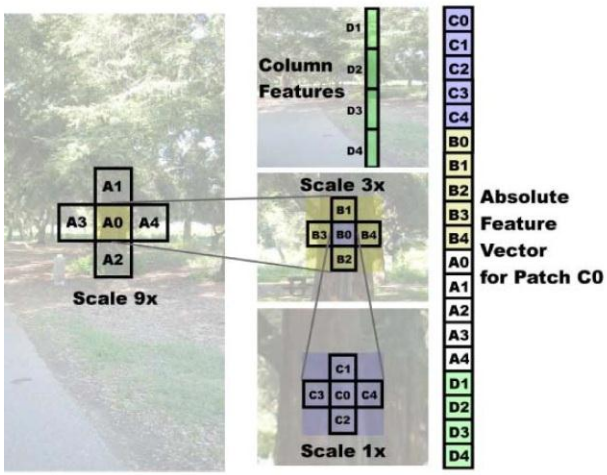

Figure 6. Showing how absolute features are made

The relative features are composed of difference histograms between neighboring pixels. The

idea here is that we compute a histogram for each patch and then compute the differences between neighboring histograms, if the patches are similar than the difference between the histograms will be smaller. This again is repeated at multiple scales.

Parameter Learning

Jointly Gaussian MRF requires us to learn 3 kinds of parameters.

1) Parameters representing the relationship between the image appearance and depth (theta)
2) Uncertainty in the ability to predict depth (sigma 1)
3) Similarity of neighboring patches in depth based on the relative features (sigma 2)

Furthermore, the idea of parameter learning is not applied in a global fashion rather parameters are learnt for each row of the depth image under the assumption that rows should have some global properties. Rows higher up in the image should represent the sky/ceiling which is similar and can be better captured by parameters rather than having a generic set of parameters which would be under fitting.

In order to learn the parameters we start by computing the parameters theta for each row modeling by simple considering dependence on image appearance.

$$d = x * \theta$$

Next we compute the sigma 1 parameters by saying

$$\sigma_{1r}^2 = v_r^T x_i \ = \mathbf{E}\ (d_i(r) - \theta_r^T x_i)^2$$
$$v_r \geq 0.$$

Now this in turn leads us to say that sigma one is actually dependent on the ability of the features to represent the relation between the depth and the features and can be estimated for each patch using its feature descriptor and the parameters *v* and so we compute the parameters *v* by solving a quadratic program so that we can constrain the values of *v* to be positive so that our sigma is always positive.

Similarly we solve for sigma two, in which case we compute the sigma2 for each row for each scale using the following relation.

$$\sigma_{2rs}^2 = u_{rs}^T |y_{ijs}| \ = \mathbf{E}(d_i(s) - d_j(s))^2$$

$$u_{rs} \geq 0$$

As the smoothness does not depend on uncertainty in ability to predict depth and is independent of

sigma1 and therefore this estimate of sigma 2 is final.

However for sigma1 and theta we iteratively refine the values by computing theta using the joint Gaussian distribution which can be represented in the following way so that at Inference time we can compute the maximum likelihood depth estimate in close form.

$$P_G(d|X;\theta,\sigma) = \frac{1}{Z_G}\exp\left(-\frac{1}{2}(d - X_a\theta_r)^T \Sigma_a^{-1}(d - X_a\theta_r)\right)$$

$$X_a = (\Sigma_1^{-1} + Q^T \Sigma_2^{-1} Q)^{-1}\Sigma_1^{-1} X$$

$$d^* = X_a\theta_r.$$

In total I have to compute the four parameters below, in order to be able to perform closed form inference at test time.

$$Q \quad \theta_r \quad u_{rs} \quad v_r$$

Current State of Project

I have implemented the features and computed the parameters of the model, and am currently removing bugs from the iterative optimization of sigma one and theta. As my current iterative approach is not working consequently I have not been able to demonstrate results for the jointly Gaussian MRF; however the results from the data term alone can be seen.

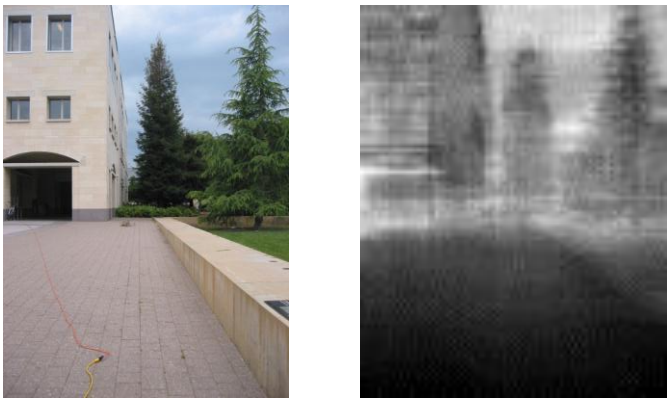

Figure 7. Reconstruction from Data term alone without iterations

This shows that the MRF is to a reasonable degree able to model the depth. Whereas the results are

visually coherent, there NCC is much lower than that for non-parametric methods.

Future Work

As soon as I figure out the bug in my code, I shall begin working on using these parameters trained on a dataset of predominantly outdoor images (Make3D) to estimate the depths for indoor images. Consequently I shall re-train the system for indoor images dataset (NYUI_V2) and evaluate the performance.

The idea behind this particular experiment is to determine whether we can have a set of global parameters that represent scene depths for multiple environments. It is expected that this will not be the case, and the parameters of the re-trained system shall outperform the parameters on different scene. Where this to be the case this would advocate that the data term should make more use of data from similar scenes then a global parameter based representation.

Consequently I intend to implement the Laplacian MRF to determine which models better the distribution of indoor scene depths.

Bibliography

[1] J. Konrad, G. Brown, M. Wang, P. Ishwar, C. Wu, and D. Mukherjee, **Automatic 2D-to-3D image conversion using 3D examples from the Internet,** in *Proc. SPIE Stereoscopic Displays and Applications*, vol. 8288, Jan. 2012

[2] - Ashutosh Saxena, Sung H. Chung, Andrew Y. Ng. **3-D Depth Reconstruction from a Single Still Image** , *International Journal of Computer Vision (IJCV)*, Aug 2007.

[3] http://cs.stanford.edu/people/asaxena/learningdepth/code/

Note : Figure 6, Figure 5 have been taken from [2]