Image Synthesis final project: Soft 3D reconstruction for view synthesis

Gael Van der Lee

Abstract

We look at the methods used in the paper “Soft 3D reconstruction for view synthesis” by Eric Penner and Li Zhang [1]. While the paper as a whole combines a lot of different techniques and can be quite complicated to reproduce in its entirety. We show that we can reproduce the depth estimation used to create the occlusion volumes and the depth maps used for image synthesis.

Introduction

Soft 3D reconstruction for view synthesis by Eric Penner and Li Zhang [1] is a paper that pertains to the field of view synthesis. View synthesis is a field of computer visions whose goal is to create new views, whether it be new camera angles or positions or both, of a given object given limited input data such as only a few images. Here, the authors use multiple images along with camera intrinsics in order to create a soft 3D representation of the layout of the picture. Soft 3D in this case means that while it does not represent the entire scene in 3D, the algorithm has an idea of the depth of the scene as well as the positional relationships between those objects, and how a different view would affect the final image.

The goal of such an algorithm would be to be able to recreate a 3D environment using sparse images of a place. Such use cases have recently been implemented in certain products. Google for example recently created an algorithm to create 3D models of touristic sites from the wealth of photos that are available on the internet [4].

Original paper

The first step is to take multiple images along with their metadata in order to compute a transformation between those images. It then uses this information in order to create one depth map per image. To create a depth map, it uses the image’s direct neighbors (2 for a 1D array of images, 4 for a 2D array of images) and uses pixel wise differences as well as an edge aware filter, a technique developed by Hosnim et al [2]. This implementation will be further detailed in the implementation section.

However, the authors found that depth estimation to be too inaccurate and chose to further refine it afterwards. So the second step is to use a discretized projective volume representation method to aggregate and store an entire depth distribution for each image

pixel. The issue that arises then is that there will be disagreement between images. The authors remedy that problem using a vote-value and vote-confidence function, where each depth map is not allowed to vote behind its depth. They also prevent noise from being amplified in the background by having the consensus algorithm fade to 0. While this method succeeds as achieving some consistency between depth maps, the authors find that achieving full consistency is unusable in practice. So they make an effort to remove depth outliers from the depth map to ensure good results. They do this by adding a small interval around the votes and then subtracting one from the votes accumulated.

The third step is then to use a similar method to calculate consensus volumes for all inputs. This volume is then used for a ray tracing algorithm which traverses the slices of the volume one by one, accumulating color on it's path, terminating when its visibility reaches 0.

A bonus step they mention is to add occlusion aware depth approximation, which further refines the depth maps using occlusion data. Though the methods only work well enough with many images, and not for simple stereo vision.

They manage to get great results, the method recreates images from new viewpoints with almost no discernable issues. The method also manages to solve complex cases such as very thin objects in the foreground. This is what encouraged me to pick this paper as we think it is fascinating and extremely impressive.

Figure 1: stereo images used as inputs for the model.

Implementation

Due to the complexity of the paper and the depth of many of the methods utilized here,We were unable to recreate it entirely. The paper relies on previous works and implementing them right proved to be a challenge.

We chose to start at the beginning and focus on the depth estimation using two images Figure 1. This covers the first of the 3 main steps described above.

The initial depth estimation used by the authors uses a method first presented in "Real-time local stereo matching using guided image filtering" by Asmaa Hosnim [2]. However, this paper relies on a filtering algorithm implemented in "Guided Image Filtering" by Kaiming He [3]. So the first step was to implement the filtering algorithm, which proved to be the most challenging part.

The paper uses what they call a *guided filter*, which uses a guide image in order to apply an edge preserving filter to the image. The filter is as follows, for each pixel i, we can express the filtered output as a weighted average:

$$q_i = \sum_j W_{ij}(I) p_j,$$

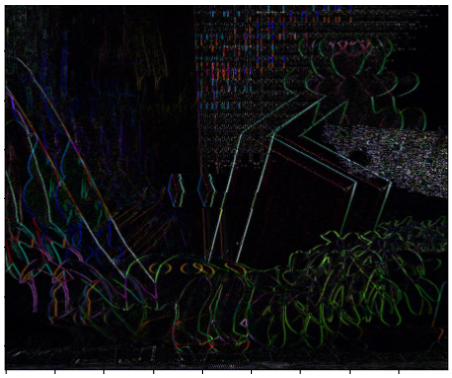
With the kernel weights are defined as follows:

$$W_{ij}(I) = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} (1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}).$$

This effectively creates a box filter that we run over the computed cost, smoothing it while still preserving the edges. We mentioned this was the most challenging part, not only because it was quite difficult to understand how to translate those equations into code that works, but also because a naive implementation can take very long to run. Thus optimization and linearization were a priority.

(1)

(2)



(3)

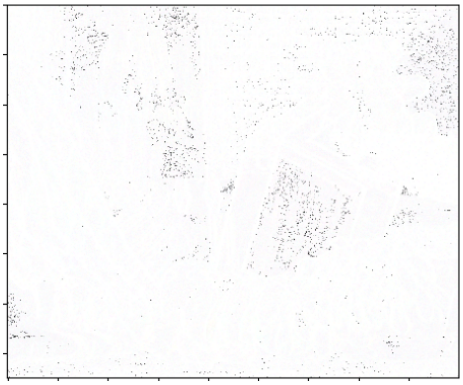


Figure 2: Intermediary outputs used for the depth map computation. (1) the SAD, (2) the gradient along x and (3) the cost.

After creating the kernel, we could use the methods from "Real-time local stereo matching using guided image filtering" by Asmaa Hosnim [2]. The way this method works is that we first compute the sum of the absolute values of the pixel wise differences across the color values between

our two images given an offset d for a pixel p. This is also called Sum of Absolute Differences or SAD. This can be more formally described as:

$$M(p,d) = \sum_{i=1}^{i=3} \left| {I^i}_{left}(p) - {I^i}_{right}(p-d) \right|.$$

The output of this step can be seen in Fig 2 (1). Then we also need for each pixel to compute the gradient of that pixel in the x direction, taking the absolute value of the difference of the gradients between our images.

$$G(p,d) = |\nabla_x(I_{left}(p)) - \nabla_x(I_{right}(p-d))|$$

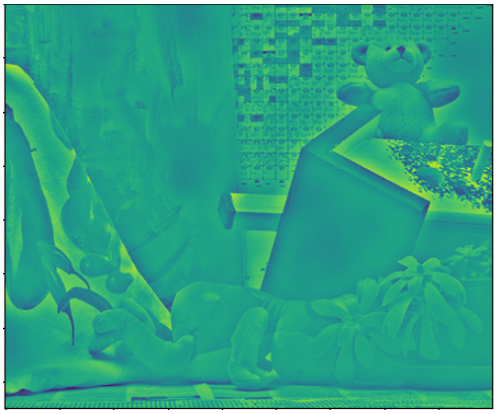
This gives us Fig 2 (2). Then we can use both the gradient and the pixel wise differences to compute an associated cost for each pixel.

$$C(p,d) = \alpha \cdot \min\left(T_c, M(p,d)\right) + (1-\alpha) \cdot \min\left(T_g, G(p,d)\right).$$

With Tc the upper limit for the differences and Tg the upper limit for the gradient. α allows us to shift the weights for one or the other. Those variables are chosen by us and could be fine tuned for better results. As you can see in Fig 2 (3), the image is mostly white, meaning that most of our values are clipped by Tc and Tg. We obtained this using the values given by the paper [2]I found that this was not an issue and thus is probably desired behavior. Finally, using the guided image filtering described above with the computed costs we can obtain the filtered cost value using:

$$C'(p,d) = \sum_q W_{p,q}(I) C(q,d).$$

(1)



(2)

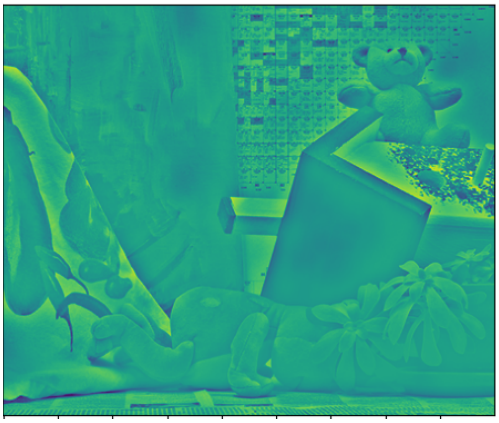


Figure 3: Filtered outputs (1) The filtered cost value, (2) The winner-takes-all final depth map

We obtain Fig 3 (1). This then goes through a winner takes all strategy, where every pixel is taken as the minimum between all depth maps, giving us the final depth map:

$$d_p = \underset{d \in \mathcal{D}}{\operatorname{argmin}}\, C'(p,d)$$

This results in our output: Fig 3 (2). As you can see, the difference between the filtered cost and the final cost is minimal. From my observations, most depth maps remain mostly similar, even when increasing d. For the visualization of the map we chose to use the viridis color palette instead of grayscale

used by the paper, because it is perceptually uniform and thus we believe it represents the underlying data better.

Those methods were initially extremely slow to compute. That was because we implemented the equations directly and were often doing per element operations. This led to each image taking minutes. However using matrix operations allowed us to speed up the computation time by multiple orders of magnitude, taking 0.125 second per image pair to compute the depth map.

Possible improvements

We hit a roadblock trying to create the soft 3D reconstruction through discretized volumes. We think that the next logical step would be to implement this method in order to be able to do soft view synthesis. Also, implementing a way to transform images through camera intrinsics would be useful. For now, we are limited to stereo images but image transformations would allow for more tan 2 images and thus more data for our estimations.

Results

We obtained those results using the same hyperparameters as [2], namely {r, ϱ, α, σs, σc, Tc, Tg} = {9, 0.0001, 0.9, 9, 0.1, 0.028, 0.008}. The results we obtained match our objectives. The depth maps are not perfect, this was expected given the criticism of the initial depth maps by the paper. We manage to capture the different objects and their outlines. The results can be seen in Figure 6.

The art studio is a good example of a good depth prediction, while the picture of the books shows a case where the model doesn't perform as well: a perfectly flat book has a sudden change of depth in the middle of it. The gifts one is interesting, the model does great on the star but also mistakes the pattern on the box for depth.

Not only do we manage to capture depth information, we also attain the optimization goal of the paper managing to make this depth estimation run in real time

Figure 6: Depth map outputs for different images

Resources

[1] Soft 3D reconstruction for view synthesis, Eric Penner Li Zhang Publication:ACM Transactions on GraphicsNovember 2017 Article No.: 235
https://doi.org/10.1145/3130800.3130855

[2] Real-time local stereo matching using guided image filtering Asmaa Hosnim, Michael Bleyer, July 2011 doi: 10.1109/ICME.2011.6012131

[3] Guided Image Filtering, Kaiming He, Jian Sun, and Xiaoou Tang, Shenzhen Institutes of Advanced Technology, June 2013, DOI: 10.1109/TPAMI.2012.213

[4] NeRF in the WildNeural Radiance Fields for Unconstrained Photo Collections, presented at CVPR 2021 (Oral)