CS284 Final project: estimate depth from monocular videos with VO and depth completion

Yintao, Xu
Shanghaitech University
`student id: 53533324`
`xuyt@shanghaitech.edu.cn`

Haomin, Shi
Shanghaitech University
`student id: 15852965`
`shihm@shanghaitech.edu.cn`

Zilin, Si
Shanghaitech University
`student id: 50696105`
`sizl@shanghaitech.edu.cn`

Abstract

Depth estimation from monocular RGB is a classical problem in computer vision. Recently, rely on boosting of deep learning, it is well-studied. However, few of state-of-art methods combine the geometric methods to refine their result. Our final project aims at estimating depth from the monocular video, which relies on combination of a newly-discussed problem in CV community: depth completion and classical methods in SLAM community: triangulation and ICP. A novel interleaving pipeline is introduced to fully make use of the benefit of geometric verification and deep-learned knowledge, with the ICP-provided dense correspondence.

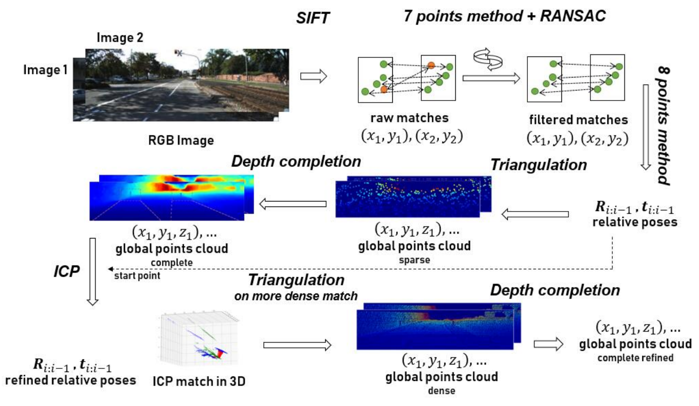

Figure 1: The whole pipeline.

1. Introduction

Depth sensing and estimation is critical in a wide range of engineering applications: augmented reality(AR), robotics, etc. Most piratical engineering of depth sensing base on extra hardware: LiDAR, kinect depth sensor. However, the power consumption and the cost circumscribes the application scenario. On some handy mobile devices, only webcam-level RGB video series is available, which calls for a powerful back-end depth estimation algorithm. Few study focus on estimate depth map from video(i.e: RGB series). Direct-VO binding with deep learning is applied in [8]. However, it does not fully utilizes the correspondence between two images.

Estimating depth from the structure of motion is the core topic of our final project. And our project aims at:

- Roughly match the two images by RANSAC and get sparse depth map.
- Complete the depth map by deep learning.
- Find dense correspondence between successive image by ICP of the depth image.
- Refine the depth map with the reconstructed correspondence.
- Construct the implementation as fast as possible.

2. Motivation and related works

The dense depth map can be gained from Lidar or Kinect sensor easily. However, the cost of them is also more than camera. We hope to realize the dense depth map with only RGB camera in real time such that with lower cost, reaching a high performance.

Since we only use the images as our input, extracting features from images and triangulating correspondences to get the depth information is a feasible way in real time.

However, sparse depth map can not provide enough information as laser sensor. Applying depth completion method can provide a more accurate dense depth estimation. Previous work has shown the power of sparse depth estimation[6], the result can be seen in figure2.

2.1. Depth completion

Depth completion problem is to estimate depth from RGB image with a subset of pixels of known corresponding depth. Different types of given pixel pattern is studied.

Sparse pixels One kind of tasks in depth completion is to estimate the depth from sparse known depth information[6] as Figure 3. Here are some works do the depth completion on the re-projected landmarks in 3D space. It is exactly what we do in our project.

Partially observed depth Due to the limitation of sensor, only partial depth image is observed[9] as Figure 4. Such completion has proved to be effective for indoor scenario, combining with the normal vectors.

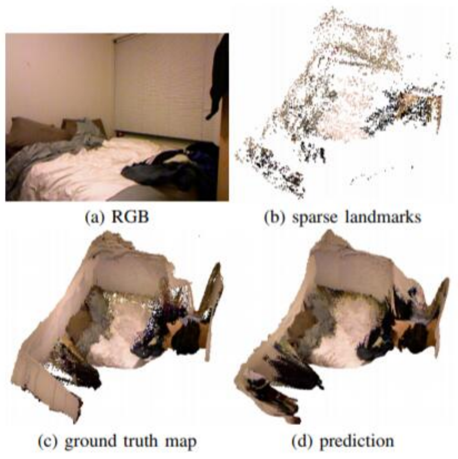

Figure 2: power of depth completion binding with visual odometry

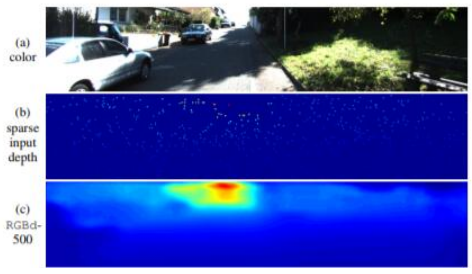

Figure 3: depth estimation bases on sparse pixels[6]

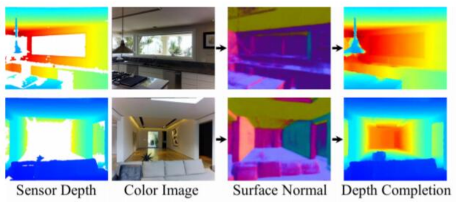

Figure 4: depth completion bases on partial observation[9]

2.2. Dense correspondence

Dense correspondence (or Scene flow) is another booming but important topic in computer vision community as Figure 5. The main task is to find pixel-level correspondence between two images between two image in many scenarios: different scene[5], instanced-based [10], geometric-supervised [2]. Benchmark like KITTI scene flow is well-built to have a good evaluation.

3. Pipeline and Implementation

The whole pipeline as Figure 1 can be summarized as:

- Find a good disparity and key frames.
- Deal with each pair of the raw RGB images. Extract features by using SIFT and filter out the mismatching outliers.
- Run RANSAC to estimate the essential matrix, then calculate the relative transformation of two frames' positions.
- Do triangulation to get a sparse depth map.
- Complete the depth map by applying neural network and output global points cloud.
- Refine the depth map by ICP matching and get the final dense depth map.

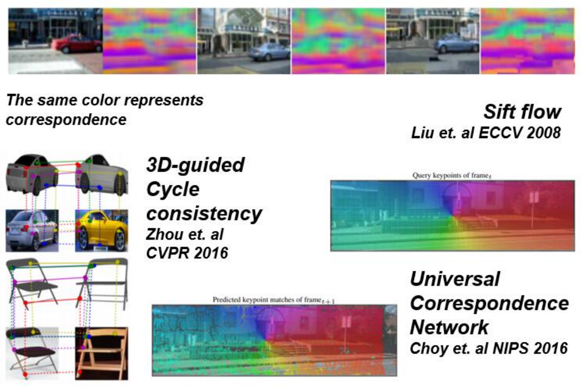
The same color represents correspondence
Sift flow
Liu et. al ECCV 2008
3D-guided Cycle consistency
Zhou et. al CVPR 2016
Universal Correspondence Network
Choy et. al NIPS 2016

Figure 5: recent dense correspondence works

3.1. Raw matches

For the first part, our goal is to build a sparse depth map only based on 2D image pair. So we applied the classical VO pipeline to realize it. Given the correspondence feature points and the relative translation between two frames, we can get the 3D point cloud and furthermore, get the sparse depth map. Since our goal was to supplant laser sensor by camera, and the laser depth information is based on the current body coordinate, the relative depth is enough.

3.2. Depth completion

In brief, the network uses sparse(or partial) depth information, combining with RGB as input, and output depth map directly, basing on knowledge in database. In our project, there should be two depth completion process.

1. Depth completion from space depth map(only landmarks available)
2. Depth completion from dense depth map(after re-triangulation from ICP correspondence).

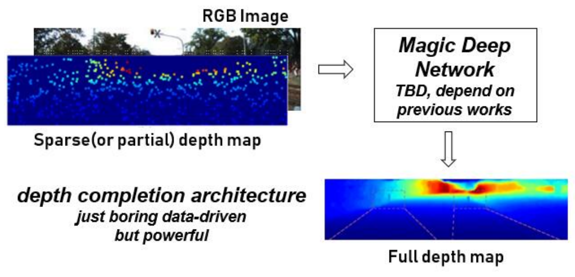
RGB Image
Sparse(or partial) depth map
Magic Deep Network
TBD, depend on previous works
depth completion architecture
just boring data-driven but powerful
Full depth map

Figure 6: input/output of depth completion

For details, there are two steps of the depth completion work.

1. Re-scale the input as close as the ground truth.
2. Do the completion of the missing depth information of other pixels(other than the landmarks).

For step (1), It sounds like the data-driven ambiguity of the scaling. But actually, from geometry, there is no way to reconstruct the scale. The very idea behind that step is the engineering experience. Deep learning network only work well under comparable scaling. Some fancy tricks like batch normalization works exactly the same way. That's why we implement the step 1.

The implementation of this step can be split into two sub steps: (1) Apply the constant factor to move the mean of all depths with respect to an image in a feasible scaling. In this experiment, we adjust it into 50. (2) Train another network to do the refinement of the rescale.

For step (2), It is fully data-driven work. For the sake of stable performance, we reproduce the similar architecture from [4]. With some components are replaced by ResNet to help convergence.

For self-completeness, I will introduce the whole architecture briefly as Figure 7. Here, I just show the architecture from that paper[4], which is the same as our architecture in general. Note that the paper is close-source now. We re-implement it by hand.

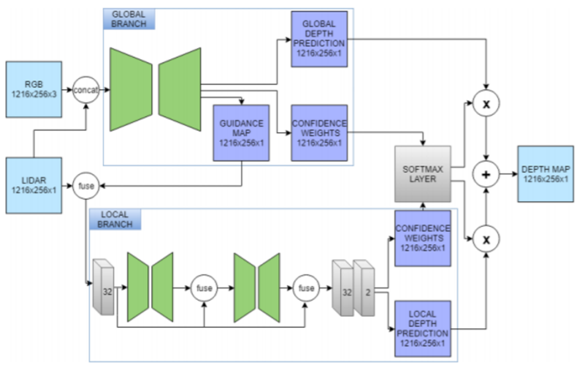

Figure 7: Scratch of architecture from[4]

The general idea is here: There are two components of this architecture: **the global branch and the local branch**. Rely on RGB and sparse depth map input, the global branch provides a confidence map, which applies to depth input to give some hints in unsupervised setting. At the same time, it also output a dense depth map and the confidence of that map for every pixel. Then, by a Resnet like recursive autoencoder in local branch, we get the refined depth map locally, also with a local confidence map at the same time. Finally, by computing the soft-max of the local and global confidence, we get a weighted sum from global dense depth map and the local one. That is the final output of our estimating result.

Training details and loss design The loss we apply is the scale-invariant RMSE from [3]. There is also a regularization which shares the inspiration from the ganimation. It is called smooth regularization. It sums up the l1-norm of the image gradient to enforce the smoothness of the output depth map. Details can refer to our code.

3.3. ICP-based correspondence search

ICP[1] is used to refine the depth map after we implement the completion of the global points cloud. Although we have a very dense depth map now, the map is blurred by the network. As we can see the first image in the Figure 8, the output of the network was separated to several blocks and the depth in a block is averaged. To solve this problem, ICP can be used to match pixels and refine the output depth clouds which can give a more accurate result.

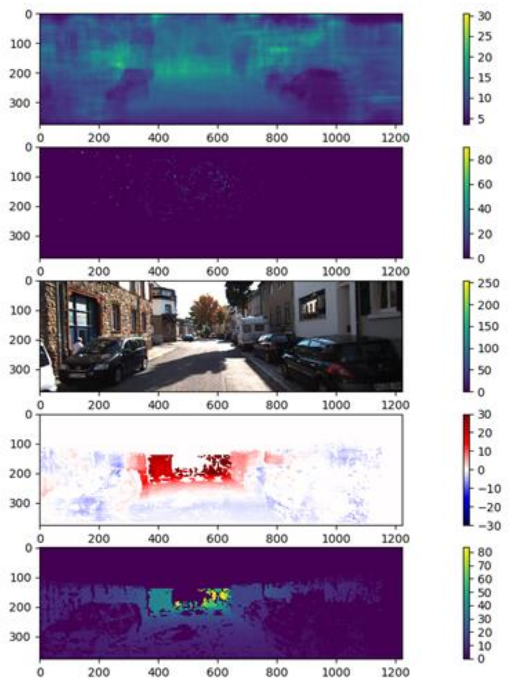

Figure 8: A sample of results.

The tricky part of this section my be the dynamic environments(especially in KITTI dataset). Since the dataset is collected outdoor, there are moving objects in the image which would affect the matching. One easy remedy(to be test) is to filter out those points are in moving or not informative(sky). Such technology has been discussed[7] and as Figure 9.

4. Usage of Codes

4.1. Prerequisite

The project is a mixture of python-3 and cpp and communicate through file system.

For python: pytorch 0.4.1, numpy, tqdm, matplotlib, etc. (you can use anaconda as one-stage solution).

For cpp: png, Eigen, OpenCV and pcl is all you need.

4.2. Running the codes

General physical Structure

triangulation: do the triangulation on image frames(RGB).

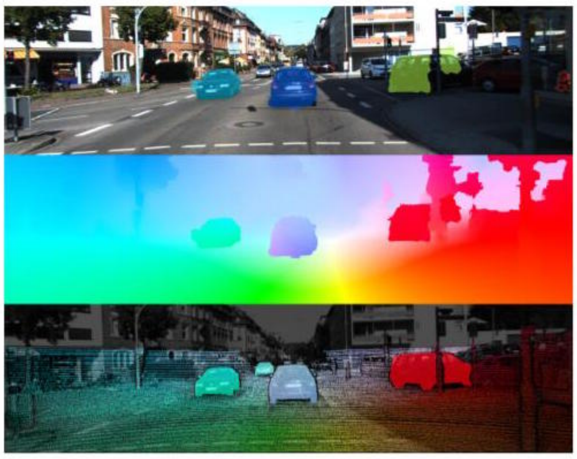

Figure 9: Current technique of masking the moving objects[7]

triangulation_evaluation: evaluate the performance of the triangualtion.

depth_completion: the deep learning model which handles the depth completion.

ICP: test the ICP and re-triangulation step on groudtruth.

To see the depth-completion result on trained model

Modify the paths in config.py to find the path of KITTI dataset

```
import os

__all__ = ["cfg"]

base_path =
    os.path.split(os.path.realpath(__file__))[0]

class cfg:
   RGB_DIR = r"/database/KITTI_RELEASED"
   D_TRAIN_DIR =
       r"/database/data_depth_annotated/train"
   D_VAL_DIR =
       r"/database/data_depth_annotated/val"
   LIST_CACHE_PATH =
       os.path.join(base_path, "data",
       "metadata")
....
```

Run *depth_completion/utils/metadata_loader.py* to preprocess the metadata.

Run *depth_completion/main_test.py* to see the result of depth completion.

5. Experiments and Results

Before experiments, we have to decide the evaluation criterion first. Thanks to [3]'s job, we have several cornerstone to evaluate the performance of the depth map.

- Threshold: % of y_i s.t. $\max(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}) = \delta < thr$
- Abs Relative difference: $\frac{1}{|T|}\sum_{y\in T}|y-y^*|/y^*$
- Squared Relative difference: $\frac{1}{|T|}\sum_{y\in T}|y-y^*|^2/y^*$

Corresponding RMSE:

- RMSE(linear): $\sqrt{\frac{1}{|T|}\sum_{y\in T}|y-y^*|^2}$
- RMSE(log): $\sqrt{\frac{1}{|T|}\sum_{y\in T}|\log y-\log y^*|^2}$
- RMSE(log, scale–invariant):

$$D(y,y^*) = \frac{1}{n}\sum(\log y_i - \log y_i^* + \alpha(y_i,y_i^*))^2 \quad (1)$$

where $\alpha(y_i,y_i^*) = \frac{1}{n}\sum_i(\log y_i - \log y_i^*)$.

5.1. depth completion network

Hyperparameters and optimizer The whole training process takes almost 4 days on a single GPU GTX1080Ti. The optimizer we use is SGD. The hyperparamters are: 0.01 as the base learning rate, with 1e-4 weight_decay and 0.9 as momentum. There is also 0.1 learning rate decay. If you have any question, you can check the documentation of the pytorch.

After implementation of the depth completion network, we first evaluated the result with KITTI ground truth based on random sampled depth map, remaining 0.1% of the total points. We tried method of Abs Relative difference with and without scale-invariance. We found out that the result always wanders around 0.999, which is hard to distinguish results. Thus we changed to the scale-invariance one. We have:

	Depth completion	State-of-art estimation
Abs Rel. Err	0.092	0.148(No prior)

5.2. triangulation

The triangulated points are evaluated by the above method too. The first method to adjust is the disparity. We have: The triangulation method did not have a good

	RMSE
disparity = 1	0.979
disparity = 5	0.430
disparity = 10	0.460

result due to several reasons. The most important one is that outdoor scene are hard to extract features and match. We have tried features of SIFT, SURF and ORB, they all worked rather poorly. We can see from 10 that the scene too far away are hard to extract features and places where intensity is too high contains no feature points at all.

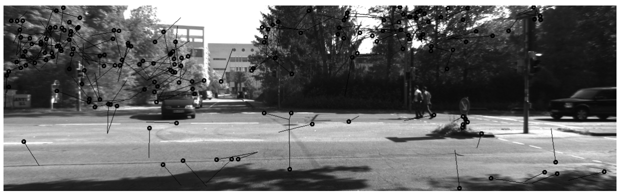

Figure 10: Feature extracted from KITTI[7]

We can see from 11 and 13 that the quality of triangulation depends greatly on the number of points extracted. If the inlier number goes below a certain number, the quality drops rapidly.

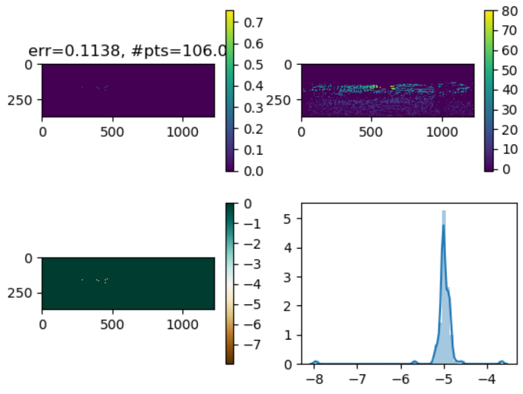

Figure 11: Result with good triangulation[7]

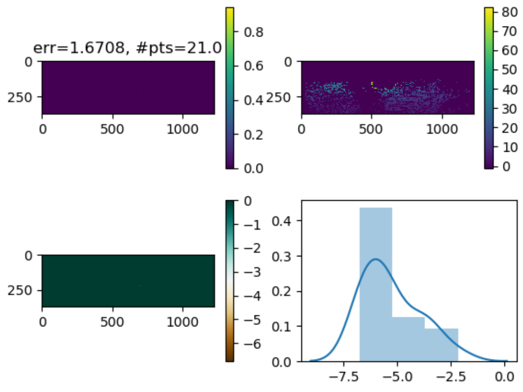

Figure 12: Result with bad triangulation[7]

Table 1: Result of Depth completion with triangulated points

	Depth completion	state of art estimation
Abs rel err	0.279	0.148

5.3. ICP and retriangulation

Here we sample points to 2% and ran ICP on ground truth. The point to plane error did not work as well as the point to point error ICP due to noise and absence of points in certain area. The ICP returns a relatively better result than the triangulation, with error reducing from 0.430 to 0.337

6. Summary and Further Work

For now, we have finished the pipeline and done the experiments on the KITTI datasets. However, the result

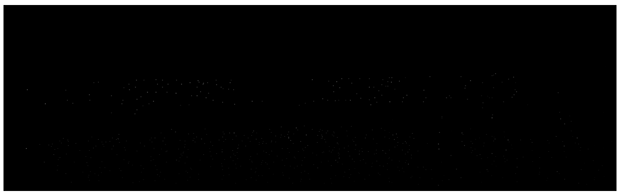

Figure 13: ICP retriangualtion result[7]

is not as good as we expect and here are several improvement that we hope to do futher. One of the problem is that the KITTI datasets are collected outdoor, and there are moving objects which would affect the performance. Also, the outdoor environment has deeper depth than indoors. The disparity in pixel intensity is also a big challenge for the pipeline to work.
But we can see that depth completion with prior knowledge of the points do give better results than pure deep learning. Even the result with poor triangulated points outperformed the traditional depth completion. Thus we are looking forward for next steps to do deeper research work on this topic.
We can first detect moving objects and remove them from the raw images, then do the matching part. This factor will influence the ICP part greatly and should be eliminated. Since our depth map is based on triangulation, the result of the larger depth may not be as good as smaller depth. We need to do more experiments on the indoor datasets, for example NYU dataset, to get rid of the depth problem and the intensity problem. Also, we consider another neural network to take the ICP result again and output a ICP based dense depth map, which might be seen as a iterative work on depth completion and caliberation.

7. Workload split scheme

It is quite challenging to implement the whole pipeline. The proper pipeline is critical. The workload split:

1. Yintao Xu: survey, paper working and deep learning section(depth completion) and presentation

2. ZiLin Si: communication between cpp(ICP, etc.) , python(deep learning framework) by well-format filesystem and report completion. It is essentially important, for depth map, it should be well-saved.

3. Haomin Shi: the more ”SLAM” works, modify and recheck ICP, triangulation, Data washing with evaluation and report completion.

References

[1] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[2] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. *CoRR*, abs/1606.03558, 2016.

[3] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *CoRR*, abs/1406.2283, 2014.

[4] W. V. Gansbeke, D. Neven, B. D. Brabandere, and L. V. Gool. Sparse and noisy lidar completion with RGB guidance and uncertainty. *CoRR*, abs/1902.05356, 2019.

[5] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994, May 2011.

[6] F. Ma and S. Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. *CoRR*, abs/1709.07492, 2017.

[7] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, June 2015.

[8] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey. Learning depth from monocular videos using direct methods. *CoRR*, abs/1712.00175, 2017.

[9] Y. Zhang and T. A. Funkhouser. Deep depth completion of a single RGB-D image. *CoRR*, abs/1803.09326, 2018.

[10] T. Zhou, P. Krähenbühl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. *CoRR*, abs/1604.05383, 2016.