Monocular Depth Reconstruction using Geometry and Deep Convolutional Networks

Ming-Fang Chang

CMU-RI-TR-18-20
April 30, 2018

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Simon Lucey, *chair*
Michael Kaess
Chen Kong, *Carnegie Mellon University*

Submitted in partial fulfillment of the requirements for the degree of Master Robotics.

Abstract

In this thesis, we explore methods of building dense depth map from monocular video using geometry and deep convolutional networks. We focus on a particular challenging case: small motion videos, in which depth error grows large as camera movement reduces.

First, we introduce our multiview stereo pipeline. Our pipeline consists of four stages: point tracking, bundle adjustment, photometric bundle adjustment, and densification. Here we demonstrate that using photometric bundle adjustment helps in getting accurate depth of textured regions from small motion video.

The traditional multiview stereo approaches rely on heuristic local smoothness priors for low-texture regions. We improve the depth estimation of low-texture region by fusing deep convolutional network predictions. We categorize the depth fusion methods into two categories: late integration and early integration. Late integration uses highly confident partial depth from pure geometric methods as anchor points to refine the dense depth map generated by deep convolutional networks. However in this case, the network output is not guaranteed to be aligned with confident partial depth, thus the fusion process might be problematic. To improve this issue, we propose early integration, which uses confident partial depths as constraints for deep convolutional networks. This method ensures the two depth sources to be well-aligned and thus has better depth accuracy than previous methods.

Acknowledgments

I would like to give sincere appreciation to all the nice people I cooperated with in CMU because these works could not be done by me alone. Thank my advisor Simon for inspiring discussions, encouragements, and all the kind supports. Thank Kit and Calvin for really great cooperation in writing papers, I learned a lot from you guys. Thank my dear labmates Ash, Chaoyang, and Chenhsuan for really warm supports and discussions. And thank my committee Michael and Chen for giving valuable suggestions to presentation and thesis writing. Finally I would like to thank my family and my cat for supporting me pursuing my interests in robot vision research.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Building depth map is a vintage problem that has been studied for many years. Depth estimation is useful for many applications such as augmented reality (measuring object size and distance), object recognition from 3D structure, robot motion planning (estimating how far should robot arm moves to reach an object), and more high-level applications.

On the other hand, high-frame-rate (HFR) camera has become a popular feature on tier-1 mobile phones in recent years. With HFR camera, we can record small motion video by shaking hand with little blurring and rolling-shutter effect. Shaking and taking small motion video takes only slight effort but can gather much more information for depth map reconstruction than taking a single-frame photo. In this thesis, we focus on depth map reconstruction using geometrical methods and deep convolutional networks.

1.1 Multiview Stereo for Small Motion Video

Reconstructing depth from small motion video is challenging because the depth error grows when camera translation decreases. To solve this problem, we propose to use photometric cues for camera pose and depth optimization, our method, as shown in the experiments, can handle wider camera baseline range and thus gives us better depth quality than previous method for small motion video.

Our multiview stereo pipeline consists of four parts[10]: point tracking, bundle adjustment(BA), photometric bundle adjustment(PBA) and densification. The point tracking step finds feature point correspondences across video sequences. And the BA step uses the point correspondences to jointly optimize camera poses and depth of feature points. The PBA step refines the result of BA using photometric cues. Finally the densification step densely searches for best depth value for each local patches[4].

1.2 Late Integration

In recent years, convolutional network has been used to predict dense depth map and surface normal map and get good quality in varies datasets[12][2]. On the other hand, our multiview stereo method, like all other purely geometric methods, fails to get accurate depth estimation in low-texture regions. Since deep networks generate dense depth map and its depth accuracy does not depend on texture as much as purely geometric method, it would be beneficial to combine the results of the two methods. To combine the advantages of multiview geometrical methods and deep neural network, we propose to use surface normal prediction by convolutional networks to fill the low-texture region where pure geometric method cannot handle. The information from surface normal prediction improves our depth map quality of low-texture regions, which was usually noisy in pure geometrical methods. The experiments using synthetic and real HFR videos show that our method can fill the missing depth in low-texture regions and outperforms previous methods in real HFR video sequences. We categorize this kind of method as "late integration", for the fact that the network part and multview stereo part are independent before depth fusion.

1.3 Early Integration

We take a forward step to think that, since we have depth from multiview stereo available, can we use it in CNN to improve network output quality? We categorize this type of method as "early integration" because it integrates sparse depth into the depth prediction process of CNN. An intuitive way to do this is to stack sparse depth to RGB image for network input like[13]. Consider that in real world environment, the density of of multiview stereo depth is ever-changing. It is hard to cover all the possible depth variations in training dataset. We propose to use sparse multiview stereo depth as constraints instead of only stacking it to network input.

To incorporate constraints in network, we propose Deep Component Analysis (DeepCA)[15]. Instead of learning a closed-form solution like traditional networks, in DeepCA framework, we decouple the link between network layers and form the inference part as a optimization problem under constraints. This structure enables us to plug in any other constraints when doing inference and thus enables us to use sparse multiview stereo depth as constraints. Our experiment shows that early integration achieves better depth accuracy than late integration and has significant improvement especially when training data is not enough.

Chapter 2

Background

Reconstructing depth map from small motion video is a an old but still challenging problem. Many Structure-from-motion (SfM) methods have been proposed through these years. By extracting and tracking features points across video frames, we can reconstruct sparse point cloud using camera geometry. ORB-SLAM[14] is a symbolic framework for large-scale sparse depth reconstruction. On the other hand, dense mapping using direct methods also draw attentions. To the best of author's knowledge, DTAM[16] is the first real-time dense depth reconstruction work using monocular camera. It minimizes photometric error across frames and use an regularization term to enforce depth smoothness in low-texture region. Direct Sparse Odometry(DSO)[7] proposed using direct pixel alignment for visual odometry, which is proven to be more accurate and more robust to photometric noise. Besides, PatchMatch Stereo[4] is proposed to solve stereo depth information with random initialization. In recent years, some papers extend PatchMatch from stereo to multiview, however, their cost function still rely on photometric information to estimate depth value thus suffer from low-texture regions[10].

2.1 Multiview Stereo Methods

Pure geometric multiview stereo (MVS) methods suffer from lacking photometric information in low-texture regions, especially in low-parallax scenario where noise and texture become more indiscernible. To overcome this problem, previous researches need locally smoothness assumptions, such as regularization term [14] or piecewise plane assumption [5], to estimate the depth of low-texture region. These methods assume that the intensity of RGB image and depth has some correlation, for example, homogeneous region in RGB image should also be homogeneous in depth maps. Yang [19] proposed depth map refinement method using intensity as guidance to smooth out noisy depth pixels, implying assumption that smooth RGB intensity leads to frontal planar depth structure. However, those assumptions are heuristic and might not hold on real objects , thus the accuracy is not guaranteed.

Most of previous works does not target at small baseline problem or throwing away low- parallax frames to avoid the well-known inaccuracy issues. In [9], Ha et al. proposed a self-calibrating bundle adjustment framework for reconstructing depth map from small motion video clip. Photometric bundle adjustment can be used to further improve the accuracy to sub-pixel level [10]. However, the raw depth map from both [9] and [10] are noisy in low-texture region, and the RGB-based depth refinement method proposed by Yang[19] used by [9] generate nice- looking but overly-smoothed, inaccurate dense depth map.

2.2 Deep Convolutional Networks for Depth Prediction

On the other hand, neural networks have been used to predict surface normal and depth map from a single RGB image in recent years. Eigen et al.[6] proposed a multiscale convolutional network for depth, surface normal, and semantic labeling prediction. Bansal et al. proposed a framework for surface normal prediction using VGG-16 convolution layers [3] in 2016, and its improved version using pixel sampling, PixelNet[2], in 2017. These works demonstrate that convolutional neural networks are capable of predicting depth and surface normal by encoding local dependencies of image pixels. According to the experiment in [8], the accuracy of single view depth estimation is better than multiview method in low-gradient area.

We believe that, instead of using heuristic smoothness assumption, the data-driven surface normal prediction or depth prediction from deep neural network are better cues for reconstructing depth in low-texture region, and can be used to fill the holes in dense depth map estimation. In [20], Zhang et al. proposed a method using surface normal prediction to for robust stereo matching. We extend this concept to multiframe instead of stereo.

All methods mentioned doesn't use partial depth information in depth prediction. As for early integration methods, as far as we know, the only related work is [13]. Ma and Karaman proposed to stack sparse depth signal to RGB image for network input. Their result shows the improvement in depth accuracy than only use RGB image. However, in reality the density of sparse depth changes a lot according to environment and camera motion, and these changes is hard to be covered in training set. In the later part of this thesis, we propose to use partial depth as constraints to solve this issue.

It is worth to mention that, although we focus on monocular method in this work, besides monocular methods, there also exists many active sensing methods for acquiring dense depth map. However, the main disadvantage of active-sensing on mobile devices is power consumption. Active sensors rely on light signals to get depth information, while the light signal cannot reach far distance or cannot compete with daylight without using large power. Large machines such as autonomous car is capable of providing this power source, while emitting light signal drains batteries on mobile devices and can only be used for short-distance applications, such as face recognition on iphone8 so far.

Chapter 3

Multiview Stereo Pipeline for Small Motion Video

In this chapter, we introduce our multiview stereo pipeline for small motion video[10]. We utilize the pipeline proposed in [10] to compute sparse depth. The process of building sparse depth consists of 4 stages: first, point tracking is performed using Kanade-Lucas-Tomasi(KLT) tracker, which localizes the corresponding feature point positions in each frames. Then bundle adjustment (BA) and photometric bundle adjustment (PBA) are used to estimate camera poses, camera intrinsics, and depths of tracked points. Finally, a multivew PatchMatch (MVPM) is used to generate dense depth map.

3.1 Point Tracking

We use ShiTomasi corner detector to extract feature point positions and use KLT tracker for point tracking. The Shi-Tomasi corner detector computes the eigen values of the Harris matrix of local image patch[17], if both eigen values are larger than a threshold, we label this patch as a corner. The KLT tracker uses spatial intensity information to direct the search for the position that yields the best match[18]. Compared with point-based feature detectors like SIFT and ORB, KLT is more suitable for small motion videos for it's sub-pixel accuracy.

We pick the first image in our small motion video as reference image and track

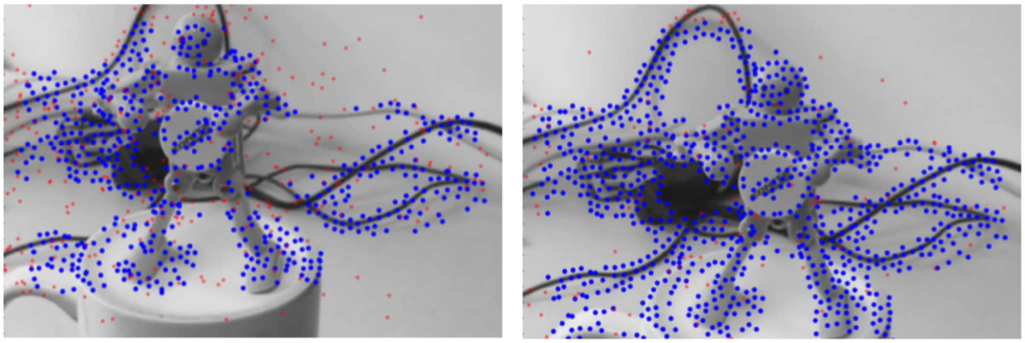


Figure 3.1: Two-frame point tracking example using KLT tracker. We use blue and red colors to represent good and failed tracked points.

the points using the local patches of first image. Second, we compute bidirectional error for each tracked point, and we judge it as failure tracking if the error is larger than 0.1 pixels. An example of point tracking result is shown in Figure 3.1.

3.2 Bundle Adjustment

With the point correspondences, we use BA to get camera poses and depth values. Notices that, instead of using triangularization method for initialization like ORB-SLAM[14], we simply initialize the depth values to be values close to one and camera poses to be small values, and then solve bundle adjustment using Gauss-Newton method. We found that this method works smoothly for small motion videos.

Consider that the camera distortion and intrinsic parameters can drift overtime and are different from module to module, besides camera poses and depth values, we also optimize camera distortion parameter and focal length. The camera distortion function α is defined as:

$$\alpha(\mathbf{x}) = 1 + k_1\|\mathbf{x}\|^2 + k_2\|\mathbf{x}\|^4 \tag{3.1}$$

where k_1 and k_2 are the first and second order radial distortion coefficients. Let f represents the focal length, $\omega, \mathbf{t} \in \mathbb{R}^3$ represents camera rotation and translation with respect to reference frame, and d represents inverse depth, the warping function from reference frame to target frame, containing camera pose parameters is defined as

$$\pi(\mathbf{x}, \omega, \mathbf{t}, d) = \langle \mathcal{R}(\omega)\mathcal{N}(\mathbf{x}) + d\mathbf{t} \rangle \,, \tag{3.2}$$

where $\mathcal{N}(\mathbf{x})$ and $\mathcal{R}(\omega)$ are defined as follows:

$$\mathcal{N}(\mathbf{x}) = \begin{bmatrix} \frac{1}{f}\alpha\left(\frac{\mathbf{x}}{f}\right)\mathbf{x} \\ 1 \end{bmatrix} \tag{3.3}$$

$$\mathcal{R}(\omega) = \begin{bmatrix} 1 & -\omega_z & \omega_y \\ \omega_z & 1 & -\omega_x \\ -\omega_y & \omega_x & 1 \end{bmatrix} \tag{3.4}$$

Here we use $\mathcal{N}$ to make homogeneous coordinates and $\mathcal{R}$ to represent rotation matrix with small angle approximation. The small angle approximation is reasonable for small motion videos.

We use angle bracket to represent the projection from 3D point to 2D image plane:

$$\mathbf{x} = \left\langle \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right\rangle = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{3.5}$$

Finally, we define the loss function for bundle adjustment. Given point correspondence across frames and initialization for camera poses and point depths, bundle adjustment minimizes reprojection error, which is the distance between originally tracked point and projected point on image. We represent the cost function of BA ,E_{BA}, by:

$$E_{\mathrm{BA}} = \sum_{i=1}^{N_f}\sum_{j=1}^{N_p} \|r_{ij}\|_{\gamma} \tag{3.6}$$

where E_{BA} minimizes point reprojection error of the N_p points across N_f frames. The error function r_{ij} is defined as:

$$r_{ij} = \mathbf{x}_{ij}\alpha(\tfrac{1}{f}\mathbf{x}_{ij}) - f\pi(\mathbf{x}_{0j}, \omega_i, \mathbf{t}_j). \tag{3.7}$$

Notice that we use huber loss $\|r_{ij}\|_\gamma$ for system robustness to outliers.

3.3 Photometric Bundle Adjustment

We use BA to generate initial camera poses, inverse depth of tracked points with respect to reference frame, camera distortion parameters, and camera intrinsics. Then perform PBA for refinement, which leads to better, sub-pixel accuracy. The cost function of photometric of E_{PBA} is represented by:

$$E_{\text{PBA}} = \sum_{i=1}^{N_f} \sum_{j=1}^{N_p} \sum_{\mathbf{x} \in \mathcal{P}_j} \|r_{ij}\|_\gamma, \tag{3.8}$$

$$r_{ij} = I_i(f\mathcal{W}(\tfrac{1}{f}\mathbf{x}_j; \Delta\omega_i, \Delta\mathbf{t}_i, \Delta d_j)) - I_0(\mathbf{x}_j), \tag{3.9}$$

where E_{PBA} minimizes the local intensity error of the points between the reference image I_0 and target frame I_i. $\mathcal{P}_j$ is the local patch around point j and $\|.\|_\gamma$ represents Huber norm. The warping function $\mathcal{W}$ is defined by camera poses:

$$\mathcal{W}(\mathbf{x}_j; \Delta\omega_i, \Delta\mathbf{t}_i, \Delta d_j) = \langle \mathcal{R}(\Delta\omega_i)\mathbf{R}_i\tilde{\mathbf{x}}_j + (\Delta d_j + d_j)(\Delta\mathbf{t}_i + \mathbf{t}_i) \rangle\,. \tag{3.10}$$

For small-motion video, PBA provides significant improvement in depth quality for following reasons:

- It can handle lower texture level than BA because it does not require every point to be a tracked feature point.
- It interpolates image patches and thus can reach sub-pixel accuracy.
- It is more robust to photometric noise as demonstrated in [7] thus is suitable for small-motion video scenario where camera motion is small so image noise is significant.

One can use direct point tracker for BA to get sub-pixel accuracy, but in this case, the image patch tracking and camera pose optimization are independent, so if a patch is not tracked accurately(for example, a patch with local 3D structure far from fronto-planar), the error in tracking cannot be fixed in BA.

In reality, optimizing all the available tracked points might make BA and PBA slow. Thus we keep the number of tracking points low in BA and PBA when doing camera pose estimation. Afterwards, we use the multiview PatchMatch described in [4] for finding inverse depth map. For example, for image sequences with resolution 640x480, we only need about 500 successfully tracked points to perform robust BA and PBA, while generally speaking, in textured environments, confident high gradient points can be numbered up to more than 10^4.

3.4 Densification

For densificatoin, we extend PatchMatch Stereo[4] to multiview form. We keep the camera poses from PBA stage and densely search for best depth value for each local patches. The best depth value is judged by photometric loss.

Searching for the best value for each patch is time- consuming. PatchMatch utilizes a belief propagation trick to accelerate the optimization process. First, we initialize the depth values randomly (the hope is that at least one pixel of the region carries a depth value that is close to the correct one). Then for local patches, we look for the depth value with least photometric error and propagate this best depth value to other pixels. In our implementation, it only take 3 to 4 iterations for the whole depth image to converge.

3.5 Experimental Results

In this experiment,we compare our method with the most recent work for small motion videos[9]. We evaluate our method and previous method using synthetic and real videos.

3.5.1 Synthetic dataset

In order to get ground truth of camera trajectory and depth map, and consider that there is no available online dataset for small motion video for now, we synthesize our own dataset.

In this experiment, we compare the quality of the depth map using a spectrum of different baselines and number of frames. In practical cases, there is no way to restrict users when taking videos, so a good system should achieve stable performance across the spectrum. With BlenderTM (ver. 2.78) we render a realistic image sequences and generated ground truth depth maps and camera poses. The resolution of our rendered image is 1280 x 720. The sequences are synthesized over a spectrum of baselines and number of frames. The motions are constant velocity with total distances ranging from 1 to 150 millimeters. Figure3.3 visualizes baselines by average images of the sequences.

Considering the real distance in common use case, we visualize the heat map to show the percentage of depth map pixels within an error of $0.015m^1$, which corresponds to less than 5cm at the mean depth in our scene (1.79m), where failure cases are represented by dark blue. Errors larger than 5cm are considered invalid.

To determine the relative depth scale to ground truth, we compute initial relative scale by minimizing least squares error between camera pose and ground truth camera pose extracted from Blender. Then we perform exhaustive search for the scale which leads to maximum number of pixels with inverse depth error less than $0.015m^1$ The result show an improved tolerance to baselines and number of frames when using our method over[9], which fails when the baseline is too small (failed to estimate from image difference) and drops performance when baseline is larger (failed to find correct point correspondences). We also note a trend that using more frames within a fixed baseline helps increase the accuracy of the depth map. For a more detailed view of the depth map errors, we visualize the cumulative error curve for two cases in Figure 3.2. We also tested OpenMVS + OpenMVG pipeline, which is a popular, open-source multiview stereo tool but is not comparable to other methods in this test. It can only cope with larger baseline and fails in most cases.

3.5.2 Real dataset

We run our method using the small motion video sequences provided by [9]. Without ground truth, it's hard to compute quantitative error. We show the depth map in Figure3.4 for visual comparison. This experiment demonstrate that our depth map has less noisy points and better overall visual quality.

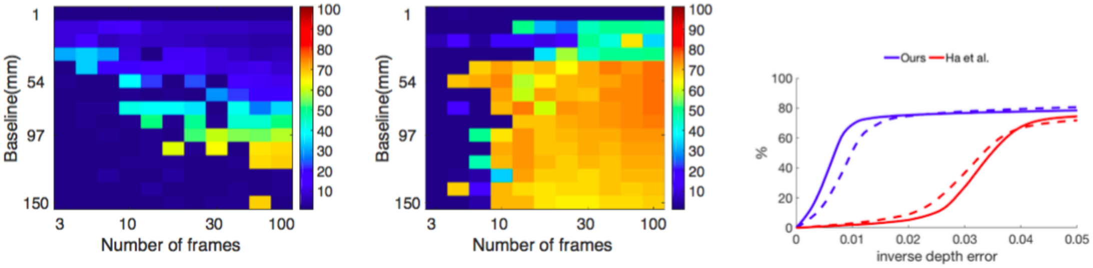
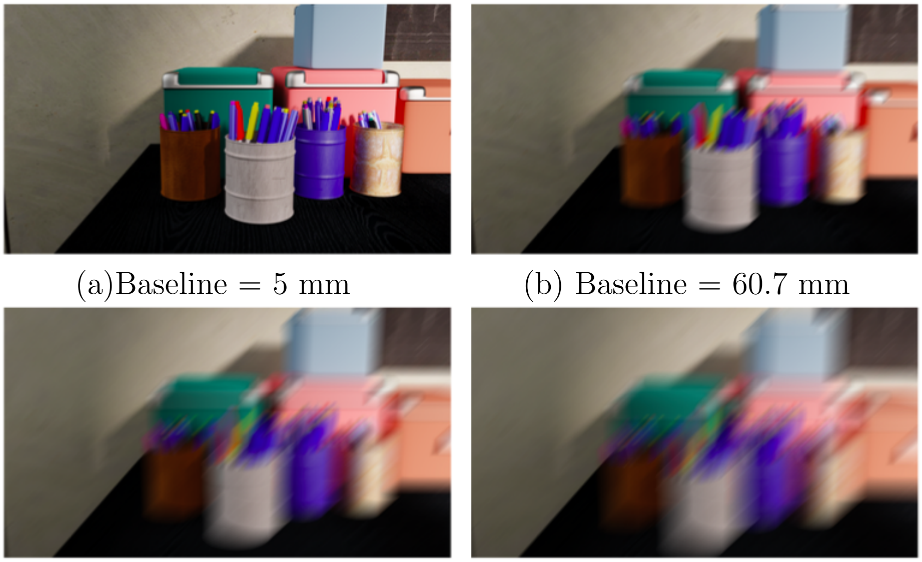


Figure 3.2: Heat map visualization for depth map accuracy. The color of heat map visualizes percentage of pixels with inverse depth error less than $0.015m^1$. Example cumulative error curves for two of the sequences show the percentage of pixels with inverse depth error less than the corresponding threshold; the solid line is for the sequence with 75mm baseline and 70 frames, the dashed line is for the sequences with 75mm baseline and 10 frames.



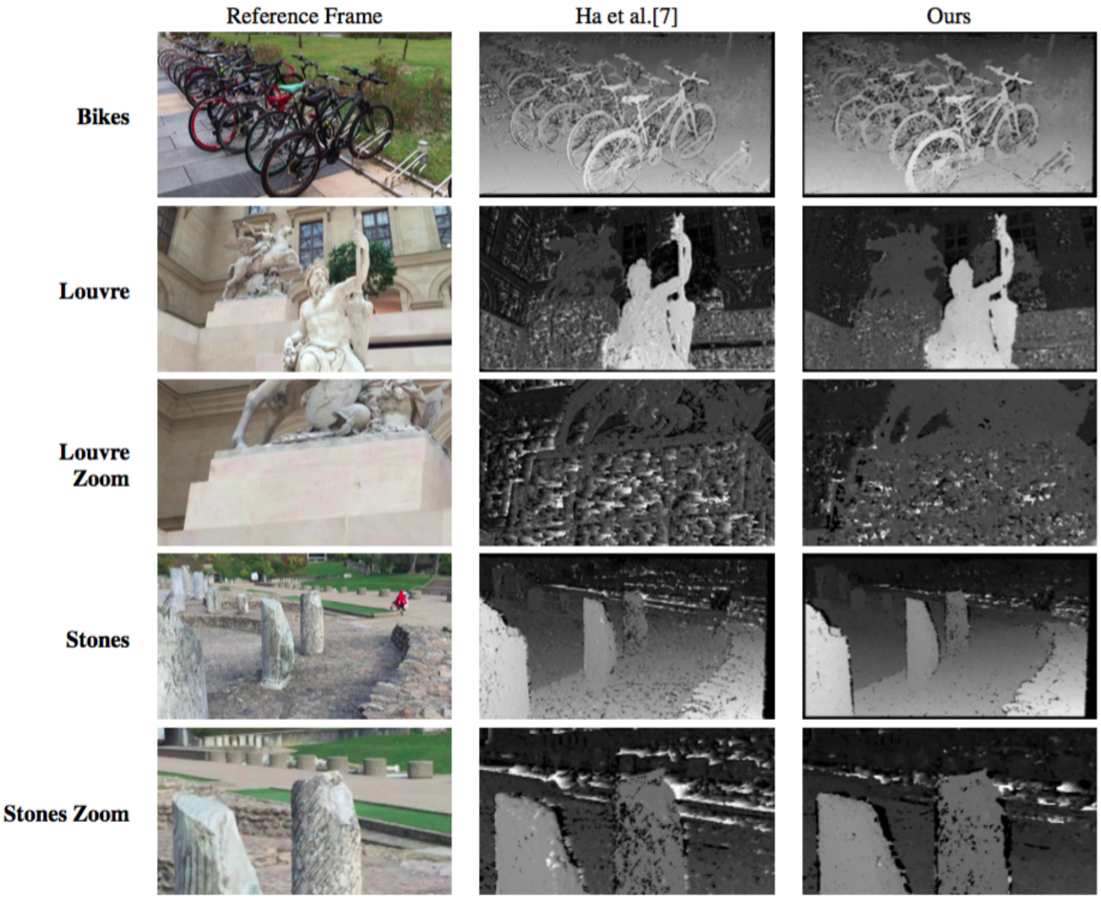
(a)Baseline = 5 mm

(b) Baseline = 60.7 mm

(c) Baseline = 130.4 mm

(d) Baseline = 200 mm

Figure 3.3: Averaged image of sequences with different baselines



(a)RGB image (b) Ha et al.[9] (c) Ours

Figure 3.4: Depth map comparison for real video sequences

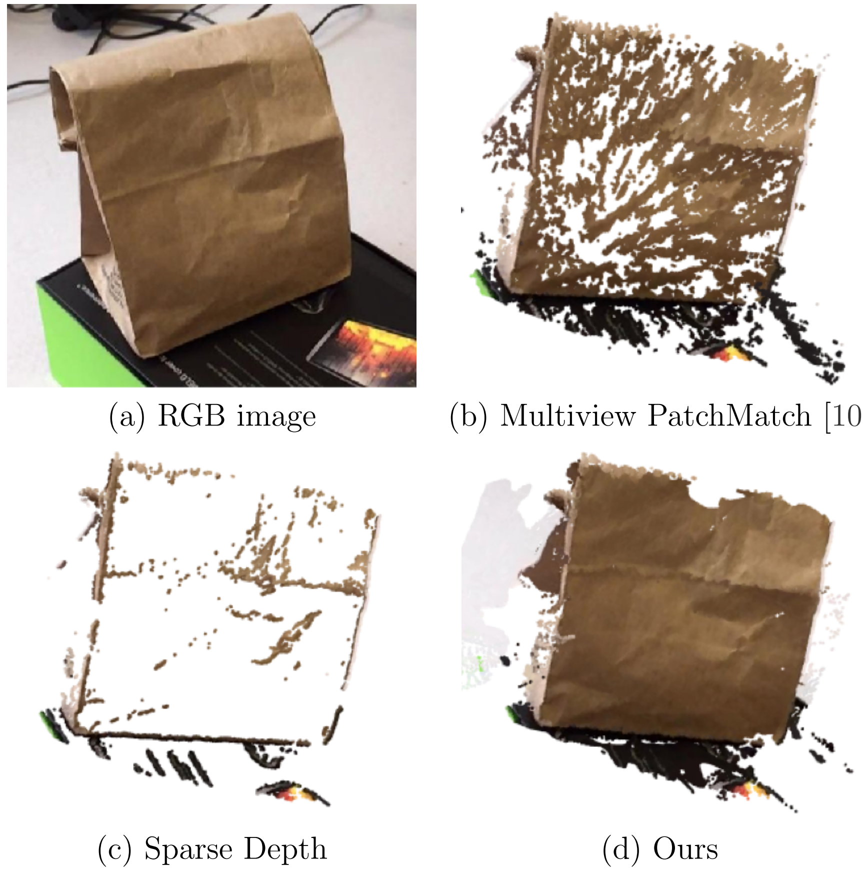
Chapter 4

Monocular Depth from Small Motion using Surface Normal Prediction

In this section, we introduce our method for generating dense depth map from small-motion videos. We use surface normal prediction from deep convolutional network to estimate depth value of low-texture region. Here we choose surface normal instead of depth as the cue for reconstruction depth map, which provides better shape details and does not need real scale to integrate. Our experiments demonstrate that, comparing with the smoothness assumption used in traditional multiview pipelines, this method achieves better overall depth accuracy. An example of our result is shown in Figure4.1.

4.1 System Overview

Our system is composed of four main modules: 1) surface normal prediction, 2) multiview stereo for sparse depth, 3) occlusion detection, and 4) final optimization process for generating dense depth map. We use the first frame of input video as reference frame in multiview stereo and as the input of surface normal prediction network. Then we reconstruct sparse depth map and occlusion mask from video sequences. We show our system block diagram in Figure 4.2.



(a) RGB image

(b) Multiview PatchMatch [10]

(c) Sparse Depth

(d) Ours

Figure 4.1: We construct dense depth map from small motion videos. Traditional methods give noisy depth estimation in low-texture regions as (b). We use surface normal prediction to fill the holes in sparse depth map in (c) and improve depth estimation accuracy in low-texture region as shown in (d)

We choose PixelNet[2] as our surface normal prediction method. It utilizes VGG-16 multilevel convolutional layers to encode multiscale image features and then form a hypercolumn descriptor. The descriptor is sent to fully-connected multilayer perceptrons to generate output class labels. In our case, the class labels represent surface normal data.

For occlusion boundary extraction, we use the method proposed in [1]. The occlusion mask is used to avoid from applying normal constraints and gradient regularization to pixels with depth discontinuity.

Our overall cost function consists of three terms: 1) orthogonality 2) anchor sparse depth points 3) gradient regularization. We introduce each term in detail in the

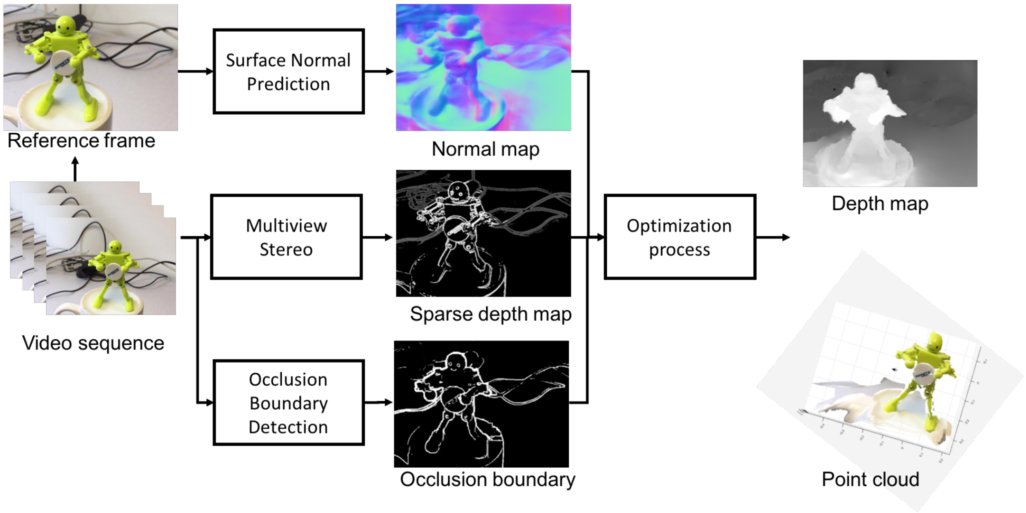


Figure 4.2: System block diagram

following sections.

4.2 Orthogonality Term

We utilize orthogonality characteristics of surface normal vectors for orthogonality term. Let $\mathbf{n}$ represents normal vector:

$$\mathbf{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix} \tag{4.1}$$

and $\mathbf{K}$ represents camera intrinsic matrix:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

where f_x and f_y stands for focal lengths and c_x and c_y are camera image center coordinate. Let $P_i(X_i, Y_i, Z_i)$ be the 3D coordinate of pixel i with reference to reference frame. The index i increases in row-major order. Following orthogonality in row direction on image, we can write:

$$\mathbf{n}^T(P_i - P_{i+1}) = 0 \tag{4.3}$$

because there is no corresponding depth difference in Y direction in 3D space along image row direction, we can write:

$$\begin{aligned} \frac{n_x}{n_z} &= -\frac{Z_i - Z_{i+1}}{X_i - X_{i+1}} \\ &= -\frac{Z_i - Z_{i+1}}{(x_i' - c_x)\frac{Z_i}{f_x} - (x_{i+1}' - c_x)\frac{Z_{i+1}}{f_x}} \end{aligned} \tag{4.4}$$

We rewrite the orthogonality term for each pixel as:

$$M_i^x Z_i - M_{i+1}^x Z_{i+1} = 0 \tag{4.5}$$

where

$$M_i^x = \frac{n_x}{f_x n_z}(x_i' - c_x) + 1 \tag{4.6}$$

In vertical direction, we follow the same way to rewrite orthogonality term as:

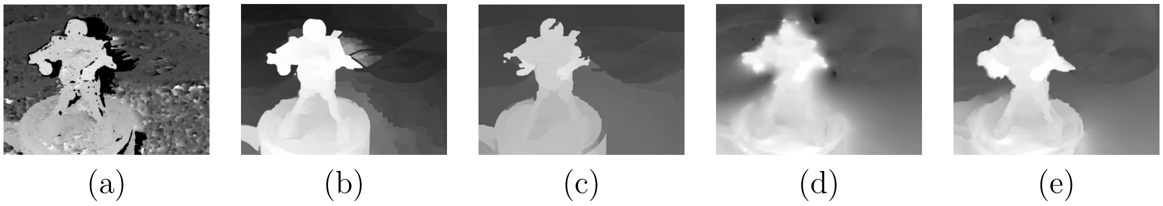
$$M_i^y Z_i - M_{i+w}^y Z_{i+w} = 0 \tag{4.7}$$

where w is image width. Then we can collect the coefficients of each pixel and write the multiplication form as follows:

$$\mathbf{M}_n \mathbf{Z} = \mathbf{0} \tag{4.8}$$

Let N be the number of pixels excluding boundaries. Notice that $\mathbf{M}$ is actually a sparse matrix with size $2N \times N$ an $\mathbf{Z}$ is a $N \times 1$ vector. The structure of $\mathbf{M}$ is:

$$\mathbf{M}_n = \begin{bmatrix} M_1^x & -M_2^x & & & & & \\ & M_2^x & -M_3^x & & & & \\ & & \vdots & & & & \\ M_1^y & & & \dots & -M_{1+w}^y & & \\ & M_2^y & & \dots & & -M_{2+w}^y & \dots \\ & & \vdots & & & & \end{bmatrix} \tag{4.9}$$



(a) (b) (c) (d) (e)

Figure 4.3: Inverse depth map comparison (a)MVPM (b) Ha et al. [9] (c) MVPM+Yang [19] (d)Naive depth reconstruction without occlusion and gradient weights (e) MVPM + Deep Normals

4.3 Gradient Regularization Term

To regularize depth gradient in potentially noisy sparse depth regions, we define another differential matrix $\mathbf{M}_g$ in similar form as $\mathbf{M}_n$

$$\mathbf{M}_g = \begin{bmatrix} 1 & -1 & & & & & \\ & 1 & -1 & & & & \\ & & \vdots & & & & \\ 1 & & & \dots & -1 & & \\ & 1 & & \dots & & -1 & \dots \\ & & \vdots & & & & \end{bmatrix} \tag{4.10}$$

The gradient regularization weighting $\mathbf{W}_g$ represents the strength of local depth smoothing. We design this term to compensate for sparse depth uncertainty. It is well-known that the uncertainty of depth estimation error is proportional to square of depth. If we use uncertain sparse depth points as anchor points, they might corrupt depth map. As a result, $\mathbf{W}_g$ is designed as $w_g w_n (1 - w_s) {Z_i}^2$ to avoid occlusion boundary and reduce strength in high gradient region, where w_g is a tuning parameter and w_n and w_s are weighting values of pixel i in W_n and W_s. We apply larger w_g when environment is planar or when noisy is larger.

4.4 Depth Optimization

Our overall cost function consists of three terms: 1) orthogonality 2) anchor sparse depth points 3) gradient regularization:

$$E_{\text{depth}} = ||\mathbf{W}_n\mathbf{M}_n\mathbf{Z}||_2^2 + ||\mathbf{W}_s(\mathbf{Z} - \mathbf{Z}_{sparse})||_2^2 + ||\mathbf{W}_g\mathbf{M}_g\mathbf{Z}||_2^2 \tag{4.11}$$

where $\mathbf{W}_n$, $\mathbf{W}_s$ and $\mathbf{W}_g$ are diagonal weighting matrices for surface normal, sparse depth, and gradient terms. The surface normal weight matrix $\mathbf{W}_n$ sets pixels on occlusion edge to zero and other pixels to 1. The sparse depth term simply enforces the depth to be aligned with sparse point from multivew stereo. The sparse depth weight $\mathbf{W}_s$ is a zero-and-one mask multiplied by a tuning weight scalar, used to select only pixels with available sparse depth for optimization.

Our result of applying occlusion mask is shown in Figure 4.3. Without occlusion mask, the foreground and background are connected and the depth discontinuity pixels are smoothed like Figure 4.3(d). After the optimization process, we apply a iterative bilateral filter to fill out missing depths for pixels in occlusion. In our implementation, we use a local variance threshold$(= 0.01)$ to filter out noisy sparse depth pixels before applying to optimization.

4.5 Experiments

We compare our method with Ha et al. [9], the most recent small motion work, and Yang [19], the depth refinement method used in [9], which uses intensity image as a guidance to refine depth map. The experiments show that our method has better performance than [9] in synthetic and real sequences. Besides, given same camera poses, our depth reconstruction method is also better than [19]. We use "MVPM" to indicate the multiview version of PatchMatch[10] . These three methods are noted as "MVPM + Deep Normals"(ours), "Ha" [9], and "MVPM + Yang" [19] in following figures.

As far as the authors know, there is no available HFR dataset for object depth reconstruction available online, or the collected video have no ground truth depth. We evaluate using the synthetic dataset provided by [9]. Besides, we collected HFR video using mobile devices with laser range finder ground truth to evaluate our method in real user application condition.

We run experiment using Macbook Air wiht 1.6GHz IntelTM Core i5 CPU and 8Gb of RAM. Total time cost depends on the choice of camera pose tracking and

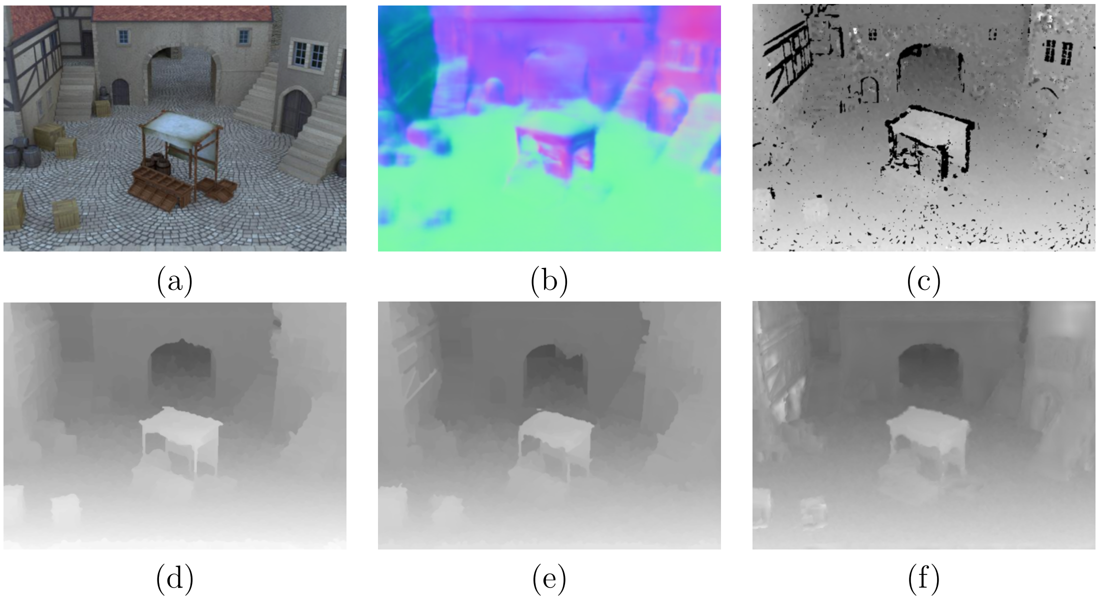


Figure 4.4: Results for synthetic dataset (a) RGB image for reference frame (b) Surface normal map (c) MVPM (d)Ha et al. [9] (e) MVPM + Yang [19] (f) MVPM + Deep Normals

occlusion methods. Without particular speedup, the time need for densification is around 30 seconds for images with 640x480 resolution.

4.5.1 Synthetic Video Clip

We evaluate our method and previous methods using the synthetic video clip provided in [9]. The input RGB image, surface normal map and inverse depth map results are shown in Figure 4.4. We can see that the inverse depth depth map of MVPM is noisy in low-texture regions. The provided ground truth is scaled to 0-255 instead of real depth. We scaled the ground truth average to 2 for better numerical comparison(which corresponds to 0.5m, a common distance when taking object video).

The accumulative error curve is shown in Fig. 4.5. The x-axis in Fig. 4.5 shows depth error range 0 to 0.1 (which corresponds to 0 10cm depth error). The y-axis shows the percentage pixels with inverse depth error less than x-axis values with respect to all valid ground truth pixels.

For this dataset, all three methods give similar accuracy in dense depth evaluation(MVPM+Yang is slightly better). This video has small baseline while is full of

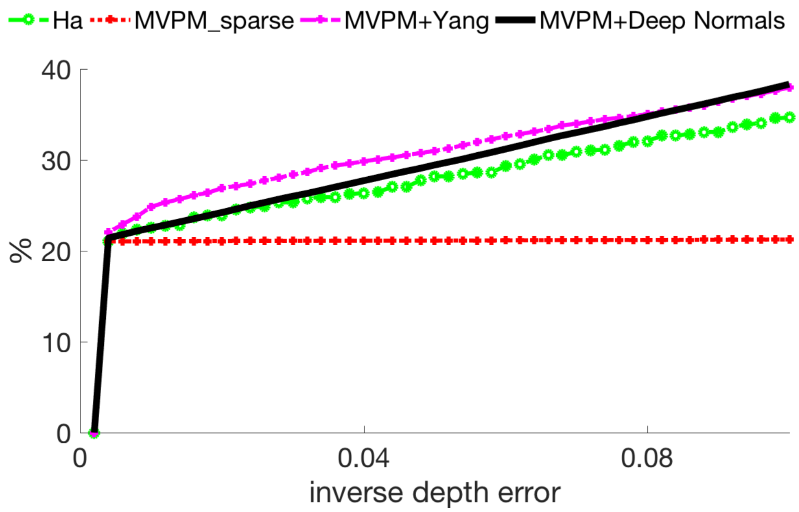


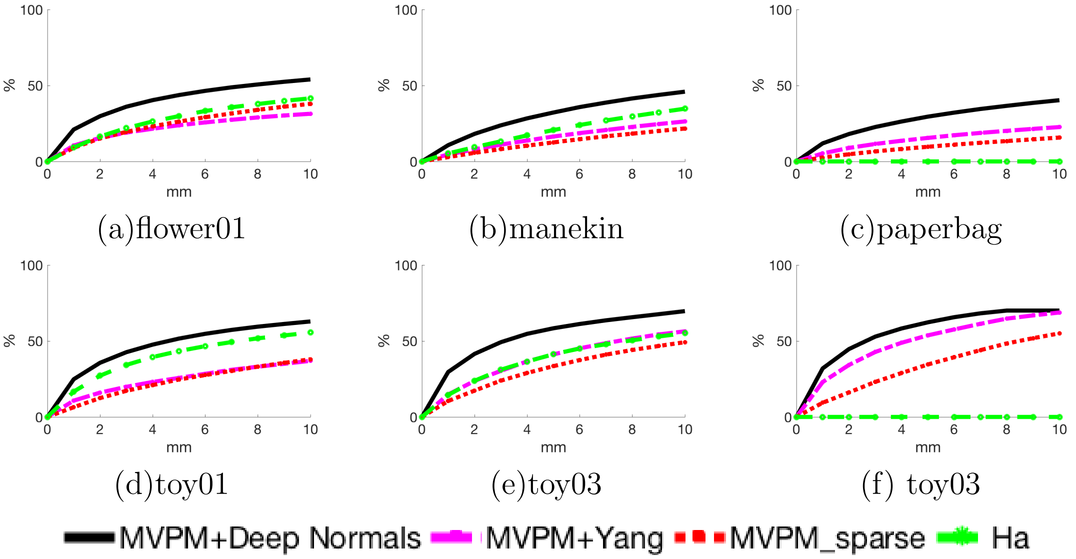
Figure 4.5: Error curves for synthetic dataset

texture, which helps all methods to estimate the sparse depth accurately for a many of pixels. The weighting parameter w_g is set to 10 in this comparison.

4.5.2 Mobile Phone Sequences

We evaluate our method using real HFR video recorded by mobile devices. We collected 6 objects with laser ground truth. The HFR videos are collected by iphone6 in 240fps, with Optical Image Stabilization(OIS) and auto focus disabled. We also use laser range finder (Konica Minolta VIVID910) to capture scans around object and perform ICP registration to generate accurate object model. The collected objects are with diverse of texture and appearance, shown in Fig. 4.7

The accmulative error curves are shown in Fig. 4.6. We perform Iterative Cloest Point(ICP) registration to align each point cloud to ground truth point cloud, with an exhaustive search for scale which minimize root mean square error of ICP inliers. The error curve are computed by the distance between ICP inliers. Higher error curve means more ICP inliers has distance less than the threshold distance in x-axis, considered as valid pixels. The y-axis shows the percentage of valid pixels with respect to all points in ground truth laser model. Here we align the depth map to one side to ground truth model. The weighting parameter w_g is set to 0.2 in this comparison.

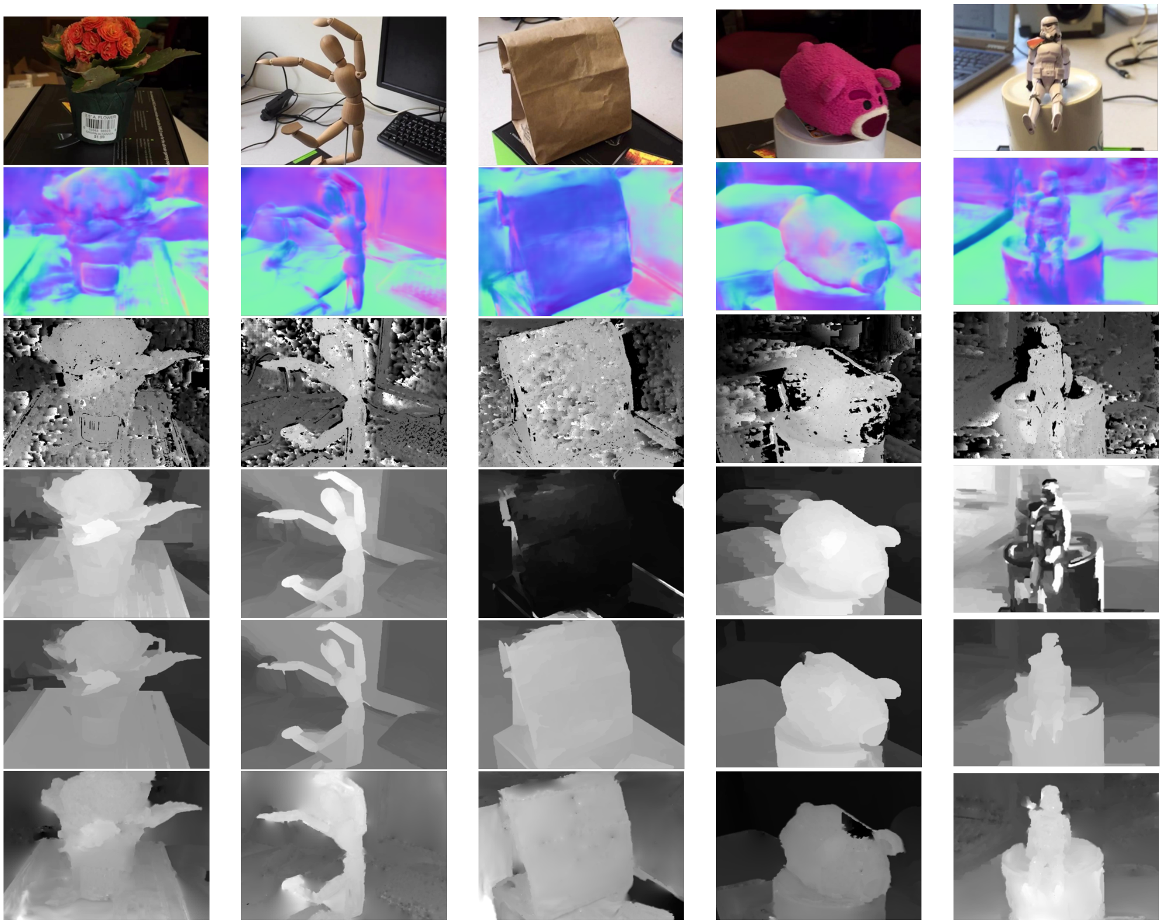


(a)flower01 (b)manekin (c)paperbag

(d)toy01 (e)toy03 (f) toy03

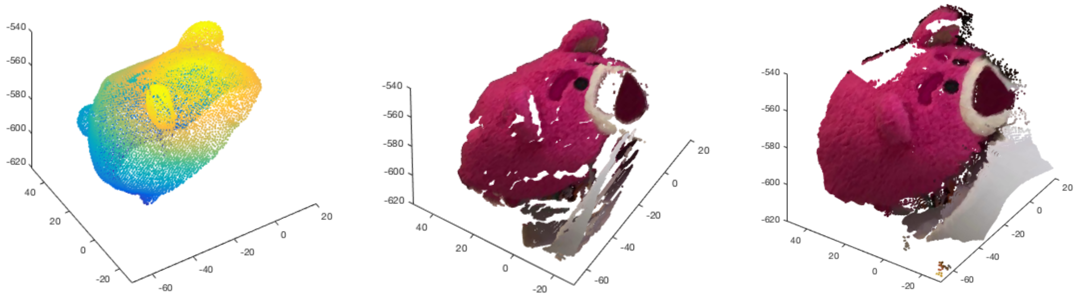
Figure 4.6: Accumulative error curves for real video sequences, the higher the better. Notice that Ha et al.[9] fails in paperbag and toy03 sequences, generating unreasonable depth estimation and unable to be aligned by ground truth, which are represented by zeros.

Visualization of inverse depth maps are shown in Fig. 5.3. We can observe that MVPM fails to construct depth in low-texture regions due to lack of photometric cue. Ha et al.'s and MVPM+Yang's method seem to generate nice depth map with sharp boundary, while the error curve is worse than our method. It is because the failure of frontal-planar assumption used in these methods, which causes over-flatten depth estimation result, as shown in Fig. 4.8.



(a) flower01 (b)manekin (c)paperbag (d)toy01 (e)toy03

Figure 4.7: Results for real sequences. From top to bottom is RGB image, surface normal map, MVPM, Ha et al [9], MVPM+Yang [19], Ours. The results of toy02 has been shown in Figure 4.2 and 4.3



(a) (b) (c)

Figure 4.8: Over-flatten depth shape caused by Yang [19] leads to less accuracy in depth map evaluation. (a) ground truth laser scan (b) MVPM+Yang [19] (c) MVPM+Deep Normals

Chapter 5

Constrained Convolutional Network

This chapter describes how we use Deep Component Analysis (DeepCA) to build constrained deep network[15]. Here we focus on DeepCA for depth reconstruction application. For more discussion about DeepCA, please refer to [15].

5.1 Deep Component Analysis

In many previous researches, people use deep neural convolutional networks to learn a closed-form solution. However, the quality of closed-form solution strongly depends on training dataset and has no guaranteed error bound. We noticed that fortunately we already have robust partial information of the network output in many cases, and the network should utilize the partial information to do inference. For multiview 3D reconstruction, we have the partial depth from lidar or multiview stereo, which are usually more robust and accurate than learnt depth map, especially in unknown environment.

We propose DeepCA to incorporate constraints in deep networks. In DeepCA formulation, a network layer output does not have to be the same as the multiplication of weighting matrix and previous layer output passed nonlinear function. Instead, it solves a constrained optimization problem for each layer. Each layer output is forced to be as close as possible to the multiplication of weighting matrix and previous layer output, under some nonlinear constraints.

One layer of feedforward network and DeepCA can be represented by:

$$\text{Feed-Forward: } \boldsymbol{a}_j = \phi(\mathbf{B}_j^{\mathsf{T}} \boldsymbol{a}_{j-1}) \implies \text{DeepCA: } \mathbf{B}_j \boldsymbol{w}_j \approx \boldsymbol{w}_{j-1} \text{ s.t. } \boldsymbol{w}_j \in \mathcal{C}_j \tag{5.1}$$

where $\boldsymbol{a}_j$ represents network layer output, $\mathbf{B}_j$ represents network parameter matrix, ϕ is nonlinear function such as ReLU or Sigmoid, $\boldsymbol{w}_j$ represents the latent variable in DeepCA which is subject to constraint $\mathcal{C}_j$. This structure enables us to plug in depth constraints to the constraint set $\mathcal{C}_j$ easily.

We can then rewrite above equation to multilayer DeepCA form of optimization problem:

$$\boldsymbol{f}^*(\boldsymbol{x}) = \underset{\{\boldsymbol{w}_j\}}{\arg\min} \sum_{j=1}^{l} \tfrac{1}{2} \left\|\boldsymbol{w}_{j-1} - \mathbf{B}_j \boldsymbol{w}_j\right\|_2^2 + \Phi_j(\boldsymbol{w}_j) \quad \text{s.t. } \boldsymbol{w}_0 = \boldsymbol{x} \tag{5.2}$$

5.2 Optimization Approach

According to [15], we can optimize our DeepCA loss function using Alternating Direction Method of Multipliers (ADMM). ADMM is an optimization method for solving complex convex optimizations, which breaks the problem into easier subproblems and solve the subproblems iteratively.

What we want to solve is:

$$\underset{\{\boldsymbol{w}_j, \boldsymbol{z}_j\}}{\arg\min} \sum_{j=1}^{l} \tfrac{1}{2} \left\|\boldsymbol{z}_{j-1} - \mathbf{B}_j \boldsymbol{w}_j\right\|_2^2 + \Phi_j(\boldsymbol{z}_j) \quad \text{s.t. } \boldsymbol{w}_0 = \boldsymbol{x}, \forall j : \boldsymbol{w}_j = \boldsymbol{z}_j \tag{5.3}$$

where $\boldsymbol{z}$ is the auxiliary variable for decoupling layer outputs and $\boldsymbol{x}$ is the network input. Rewriting the constraints using Lagrange multiplier, the loss function becomes:

$$\mathcal{L}_\rho = \sum_{j=1}^{l} \tfrac{1}{2} \left\|\boldsymbol{z}_{j-1} - \mathbf{B}_j \boldsymbol{w}_j\right\|_2^2 + \Phi_j(\boldsymbol{z}_j) + \boldsymbol{\lambda}_j^{\mathsf{T}}(\boldsymbol{w}_j - \boldsymbol{z}_j) + \tfrac{\rho}{2} \left\|\boldsymbol{w}_j - \boldsymbol{z}_j\right\|_2^2 \tag{5.4}$$

We then perform ADMM to optimize this loss funcion. The ADMM for DeepCA has following steps:

1. Update $\boldsymbol{w}_j$:

$$\begin{aligned} \boldsymbol{w}_j^{[t+1]} & \arg\min_{\boldsymbol{w}_j} \mathcal{L}_\rho(\boldsymbol{w}_j, \boldsymbol{z}_{j-1}^{[t+1]}, \boldsymbol{z}_j^{[t]}, \boldsymbol{\lambda}_j^{[t]}) \\ & = \left(\mathbf{B}_j^\intercal \mathbf{B}_j + \rho \mathbf{I}\right)^{-1} (\mathbf{B}_j^\intercal \boldsymbol{z}_{j-1}^{[t+1]} + \rho \boldsymbol{z}_j^{[t]} - \boldsymbol{\lambda}_j^{[t]}) \end{aligned} \tag{5.5}$$

2. Update $\boldsymbol{z}_j$:

$$\begin{aligned} \boldsymbol{z}_j^{[t+1]} & \arg\min_{\boldsymbol{z}_j} \mathcal{L}_\rho(\boldsymbol{w}_j^{[t+1]}, \boldsymbol{w}_{j+1}^{[t]}, \boldsymbol{z}_j, \boldsymbol{\lambda}_j^{[t]}) \\ & = \phi_j\big(\tfrac{1}{\rho+1}\mathbf{B}_{j+1}\boldsymbol{w}_{j+1}^{[t]} + \tfrac{\rho}{\rho+1}(\boldsymbol{w}_j^{[t+1]} + \tfrac{1}{\rho}\boldsymbol{\lambda}_j^{[t]})\big) \\ \boldsymbol{z}_l^{[t+1]} & \phi_j\big(\boldsymbol{w}_j^{[t+1]} + \tfrac{1}{\rho}\boldsymbol{\lambda}_j^{[t]}\big) \end{aligned} \tag{5.6}$$

3. Update $\boldsymbol{\lambda}_j$:

$$\boldsymbol{\lambda}_j^{[t+1]} \boldsymbol{\lambda}_j^{[t]} + \rho(\boldsymbol{w}_j^{[t+1]} - \boldsymbol{z}_j^{[t+1]}) \tag{5.7}$$

Notice that although the original ADMM repeats this process until convergence, in our implementation, we fix the number of iteration, and implement the weight multiplications as convolutional and transposed convolutional layers. In this way the DeepCA optimization process becomes a recursive network, which can simply be trained using backward propagation like traditional feed-forward networks.

5.3 Experiments

We demonstrate the result applying DeepCA with ADMM iterations for full-size depth map prediction. In [13], Ma and Karaman have shown that stacking sparse depth information to RGB image as network input can achieve better depth prediction performance than only using RGB.

We aim to demonstrate the advantage of using sparse depth information as constraints. Here we subsampled ground truth depth map to simulate sparse depth information. In practical cases, the sparse depth can be acquired from lidar or

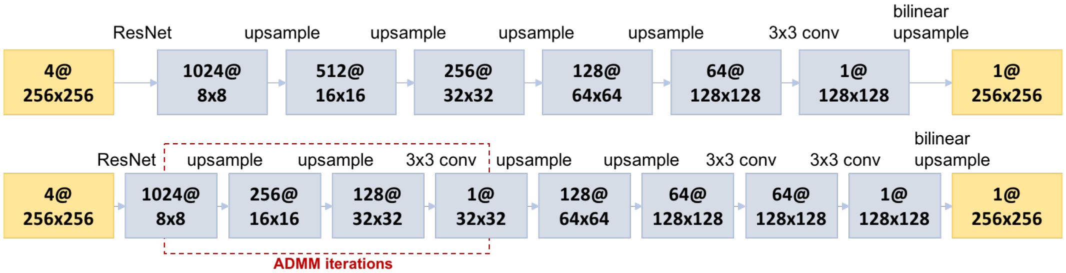


Figure 5.1: Network Structures. We adopt an encoder-decoder structure like [13], which consists of using ResNet[11] encoders and upsampling layers. The upper block diagram shows our baseline, using the same upsampling layers as [13], and the lower layer shows our DeepCA network structure.

structure-from-motion algorithms whose depth measurements are usually sparse but more accurate and robust to unknown environmental conditions. We compare our method with [13], which contains a ResNet [11] encoder and upsampling layers as decoders. The upsampling layers consist of four transposed convolutional layers, one 3x3 convolutional layer ,and a bilinear upsampling layer. To apply constraints to decoders, we first subsampled the sparse depth map from 256x256 to 32x32 using max-pooling, running ADMM iterations at 32x32 resolution scale, and then apply upsampling layers to go back to normal size. We found that, instead of doing ADMM iterations on full-size images, this strategy helps to eliminate discontinuities around sparse constraints and is faster to train. We use ResNet-18 and ResNet-10 as encoders (our ResNet-10 is the ResNet-18 without repeating each convolution blocks [11]). Our network structures are shown in Figure 5.1.

We use NYUDepthV2 dataset for evaluation. For testing set, we use the official split, which contains 654 images. For training set, we first sampled images with even interval from NYUDepthV2 raw data, skipping the categories in testing set, and randomly selected 60000 and 1000 images as training datasets. All ground truth depth maps are pre-processed using official tool box to fill in missing pixels. Afterwards, we randomly sampled 200 points from ground truth depth map to simulate sparse depth input. For all experiments, we use batch size=8 and learning rate 3×10^{-5}, which decreases to 1×10^{-5} after 20_{th} epoch, 3×10^{-6} after 80_{th} epoch, and 1×10^{-7} after 150_{th} epoch. Our platform is a 28-core Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz machine with ubuntu 16.04 and GeForce GTX TITAN X graphics cards.

Table 5.1: Using ADMM iterations to fix constraints

Item	Param	RMSE	Rel	δ_1	δ_2	δ_3
Resnet18	1.5×10^7	0.541	0.159	79.2	94.7	99.4
Resnet18 (T = 10)	1.2×10^7	0.277	0.061	95.5	99.4	99.9
Resnet10	8.8×10^6	0.555	0.157	79.8	94.6	99.4
Resnet10 (T = 10)	6.5×10^6	0.235	0.045	**97.3**	**99.6**	**99.9**
Resnet50 [13]	3.4×10^7	**0.230**	**0.044**	97.1	99.4	99.8

The error metrics are listed as follows [13]:

$$RMSE = \sqrt{\frac{1}{N}\sum_i (d_i - g_i)^2}$$

$$Rel = \frac{1}{N}\sum_i \frac{|d_i - g_i|}{g_i}$$

where g_i represents ground truth depth value and d_i represents network output. And δ^i is the percentage of predicted pixels where the larger relative error in both directions is within a 1.25^i. We rescaled our results back to the NYUDepthV2 resolution (480x640) before comparison.

5.3.1 Applying Sparse Output Constraints

We compare the depth prediction result with and without sparse output constraint. For this experiment, the models are trained using 60000 training images for 52 epochs. The results are shown in Table 5.1 and Figure 5.2 . Notice that our networks sizes for NYUDepth-V2 are much smaller than [13]'s, which uses ResNet-50 instead.

5.3.2 Train with insufficient data

In this experiment, we show that using sparse depth as constraints has significant advantage when training data is insufficient. In this case, fixing constraints (which is actually known information and holds in both training and testing) helps to significantly improve the result. This implies that our method should be more robust in unknown environments. The models are trained using 1000 images for 296 epochs.

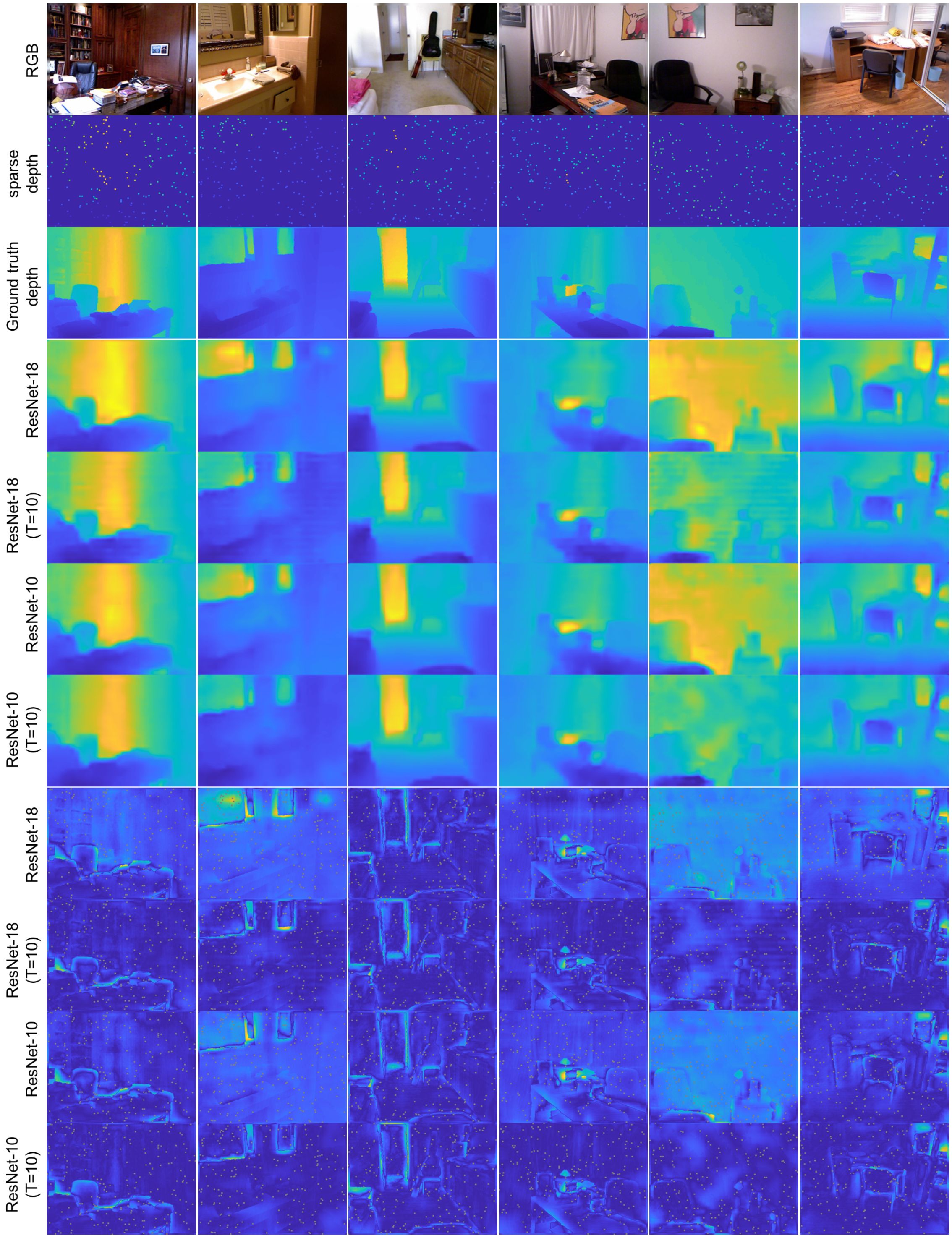


Figure 5.2: Visualization of results(trained with 60000 samples). From top to bottom are: RGB image, sparse depth , ground truth depth map, predicted depth maps, and absolute predicted depth error maps, which are overlaid with location of sparse depths(gray points). From the results, we can observe that fixing sparse constraints helps to minimize error values in the neighborhood of sparse constraints. Notice that depth error maps are normalized and sparse point size are enlarged for clear visibility.

Table 5.2: Using insufficient training data (1000 samples)

Item	Param	RMSE	Rel	δ_1	δ_2	δ_3
Resnet10	8.8×10^6	0.820	0.292	55.9	85.2	96.0
Resnet10 (T = 10)	6.5×10^6	0.545	0.172	76.9	94.8	98.8
Resnet10 (T = 20)	6.5×10^6	**0.348**	**0.081**	**92.7**	**98.6**	**99.7**

Our results are shown in Table 5.2 and Figure 5.3 .

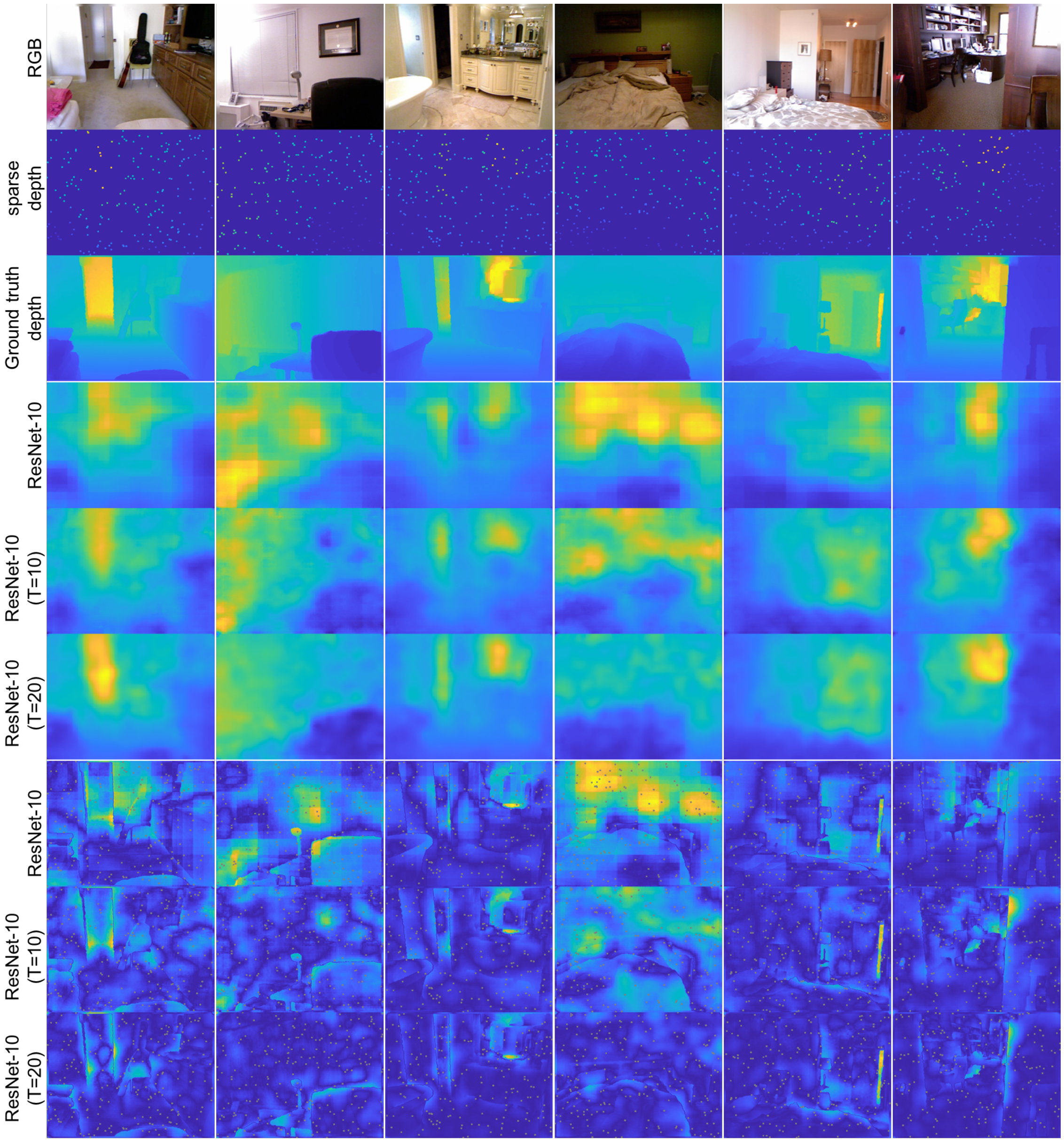


Figure 5.3: Visualization of results using insufficient training data (1000 samples). From top to bottom are: RGB image, sparse depth, ground truth depth map, predicted depth maps, and absolute predicted depth error maps, which are overlaid with location of sparse depths(gray points). Without constraints, even though the sparse depth values are stacked to network input, predicted depth map does not align with sparse depth, especially when training data is insufficient. Notice that depth error maps are normalized and sparse point size are enlarged for clear visibility.

Chapter 6

Conclusions

In this thesis, we look into the depth reconstruction problem from small motion video. We examined the traditional multiview stereo method and the usage of deep convolutional priors, including late and early integration methods. We proposed DeepCA, a early integration method which enables us to use sparse depth as constraint for deep networks. It achieves better depth accuracy and robustness than previous late integration method.

We believe that there are still many open questions and possibilities in this field. We are interested in examining failure cases of multiview stereo to make our depth fusion method more robust in real environments. In addition, here we only used sparse depth as constraints, while there are many other possible constraints to explore.

Bibliography

[1] Alper Ayvaci, Michalis Raptis, and Stefano Soatto. Occlusion Detection and Motion Estimation with Convex Optimization. *Advances in Neural Information Processing Systems*. URL https://papers.nips.cc/paper/4118-occlusion-detection-and-motion-estimation-with-convex-optimization.pdf. 4.1

[2] Aayush Bansal, Xinlei Chen, Bryan Russell, Abhinav Gupta, and Deva Ramanan. PixelNet: Towards a General Pixel-Level Architecture. URL https://arxiv.org/pdf/1609.06694.pdf. 1.2, 2.2, 4.1

[3] Aayush Bansal, Bryan Russell, and Abhinav Gupta. Marr Revisited: 2D-3D Alignment via Surface Normal Prediction. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, 2016. URL https://arxiv.org/pdf/1604.01347.pdf. 2.2

[4] Michael Bleyer, Christoph Rhemann, and Carsten Rother. PatchMatch Stereo -Stereo Matching with Slanted Support Windows. *Bmvc. Vol. 11.*, 2011. URL https://www.microsoft.com/en-us/research/wp-content/uploads/2011/01/PatchMatchStereo{_}BMVC2011{_}6MB.pdf. 1.1, 2, 3.3, 3.4

[5] Alejo Concha, Wajahat Hussain, Luis Montano, and Javier Civera. Manhattan and Piecewise-Planar Constraints for Dense Monocular Mapping. *Robotics: Science and Systems*, 2014. URL http://webdiis.unizar.es/{~}jcivera/papers/concha{_}etal{_}rss14.pdf. 2.1

[6] David Eigen and Rob Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. *Proceedings of the IEEE International Conference on Computer Vision*, 2015. URL https://arxiv.org/pdf/1411.4734.pdf. 2.2

[7] Jakob Engel and Vladlen Koltun. Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. URL https://vision.in.tum.de/{_}media/spezial/bib/engel{_}et{_}al{_}pami2018.pdf. 2, 3.3

[8] José M Fácil, Alejo Concha, Luis Montesano, and Javier Civera. Single-View and Multi-View Depth Fusion. *IEEE Robotics and Automation Letters 2.4 (2017):*

1994-2001., 2017. doi: 10.1109/LRA.2017.2715400. URL http://ieeexplore.ieee.org/document/7949041/. 2.2

[9] Hyowon Ha, Sunghoon Im, Jaesik Park, and Hae-Gon Jeon. High-quality Depth from Uncalibrated Small Motion Clip. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. URL https://www.cv-foundation.org/openaccess/content{_}cvpr{_}2016/papers/Ha{_}High-Quality{_}Depth{_}From{_}CVPR{_}2016{_}paper.pdf. (document), 2.1, 3.5, 3.5.1, 3.5.2, **??**, 4.3, 4.5, 4.4, 4.5.1, 4.6, 4.7

[10] Christopher Ham, Ming-Fang Chang, Simon Lucey, and Surya Singh. Monocular Depth from Small Motion Video Accelerated. *3D Vision*, 2017. URL http://www.ci2cv.net/media/papers/3dv{_}fastdepth.pdf. 1.1, 2, 2.1, 3, **??**, 4.5

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition.*, 2016. URL https://arxiv.org/pdf/1512.03385.pdf. (document), 5.1, 5.3

[12] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper Depth Prediction with Fully Convolutional Residual Networks. *3D Vision (3DV), 2016 Fourth International Conference on. IEEE*, 2016. URL https://arxiv.org/pdf/1606.00373.pdf. 1.2

[13] Fangchang Ma and Sertac Karaman. Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image. *Robotics and Automation (ICRA), 2018 IEEE International Conference on. IEEE*, 2018. URL https://arxiv.org/pdf/1709.07492.pdf. (document), 1.3, 2.2, 5.3, 5.1, 5.1, 5.3, 5.3.1

[14] Raúl Mur-Artal, J M M Montiel, and Juan D Tardós. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics 31.5 (2015): 1147-1163.*, 2015. doi: 10.1109/TRO.2015.2463671. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7219438. 2, 2.1, 3.2

[15] Calvin Murdock, Ming-Fang Chang, and Simon Lucey. Deep Component Analysis via Alternating Direction Neural Networks. URL https://arxiv.org/pdf/1803.06407.pdf. 1.3, 5, 5.2

[16] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. DTAM: Dense Tracking and Mapping in Real-Time. *Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE*, 2011. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.445.1843{&}rep=rep1{&}type=pdf. 2

[17] Jianbo Shi and Carlo Tomasi. Good Features to Track. 1993. URL https://users.cs.duke.edu/{~}tomasi/papers/shi/TR{_}93-1399{_}Cornell.pdf. 3.1

[18] Carlo Tomasi and Takeo Kanade. Shape and Motion from Image Streams: a Factorization MethodPart 3 Detection and Tracking of Point Features. 1991. URL https://cecas.clemson.edu/{~}stb/klt/tomasi-kanade-techreport-1991.pdf. 3.1

[19] Qingxiong Yang. A Non-Local Cost Aggregation Method for Stereo Matching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012. (document), 2.1, 4.3, 4.5, 4.4, 4.7, 4.8

[20] Shuangli Zhang, Weijian Xie, Guofeng Zhang, Hujun Bao, and Michael Kaess. Robust Stereo Matching with Surface Normal Prediction. *Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE*, 2017. URL https://frc.ri.cmu.edu/{~}kaess/pub/Zhang17icra.pdf. 2.2