Contents

1 Introduction

1.1 Recovering 3D Geometry from Images

Traditional multi-view reconstruction approaches use hand-crafted similarity metrics Like (e.g. NCC) and regularizations techniques (SGM [6]) to recover 3D points. It is reported in recent MVS benchmarks [1, 8] that, although traditional algorithms [2, 3, 10] perform very well on the *accuracy*, the reconstruction *completeness* still has large room for improvement. The main reason for the low completeness of traditional methods is because the hand-crafted similarity measure and block matching method mainly works well with Lambertian surfaces and fail in the following failure cases:

- **Textureless Surfaces** It is hard to infer the geometry from the textureless surface (e.g. white wall) since it looks similar from different viewpoints.

- **Occlusions** The scene objects may be partly or wholly invisible in different views due to the scene occlusions.

- **Repetitions** Block matching techniques can give a similar response to different surfaces due to their geometric and photometric repetitiveness.

- **Non-Lambertian Surfaces** Non-Lambertian surfaces look different from the different viewpoints.

- **Other non-geometric variations:** image noise, vignetting effect, exposure change, and lighting variation.

Zbontar et al. [15] have shown that doing block matching on feature spaces can give more robust results and can be used for depth perception in a two-view stereo setting. The goal of Multi-view stereo techniques is to estimate the dense representation from overlapping calibrated views. Recent learning-based MVS methods [11, 13, 14] were able to get more complete scene representations by learning the depth maps from feature space.

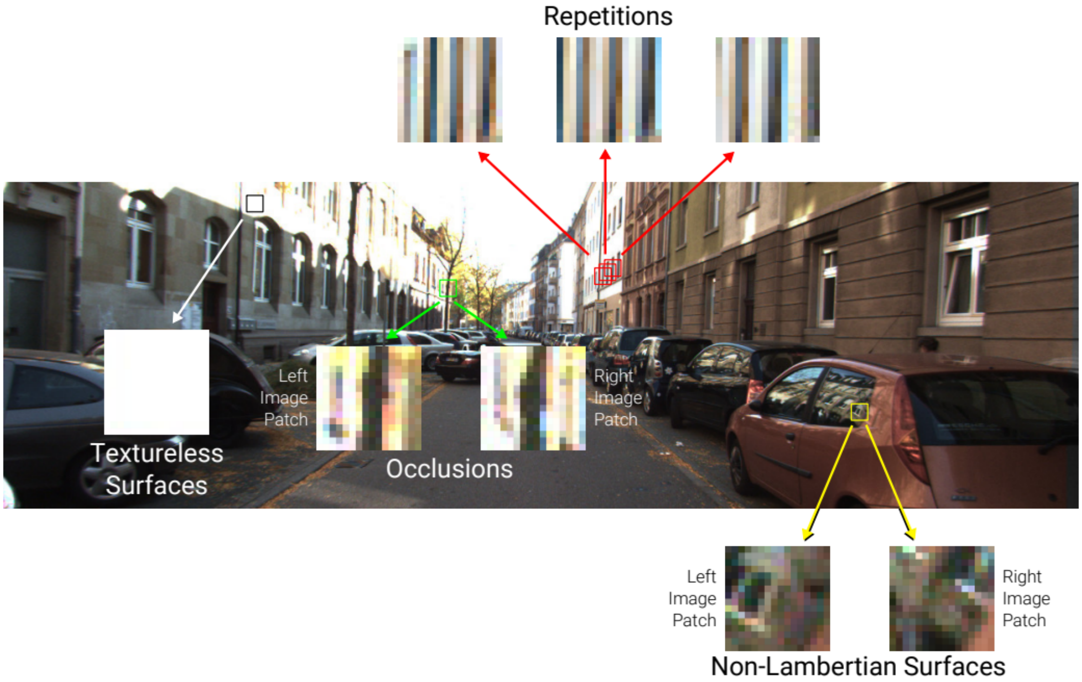

Figure 1.1: Failure cases of block matching. (Image credit: Andreas Geiger)

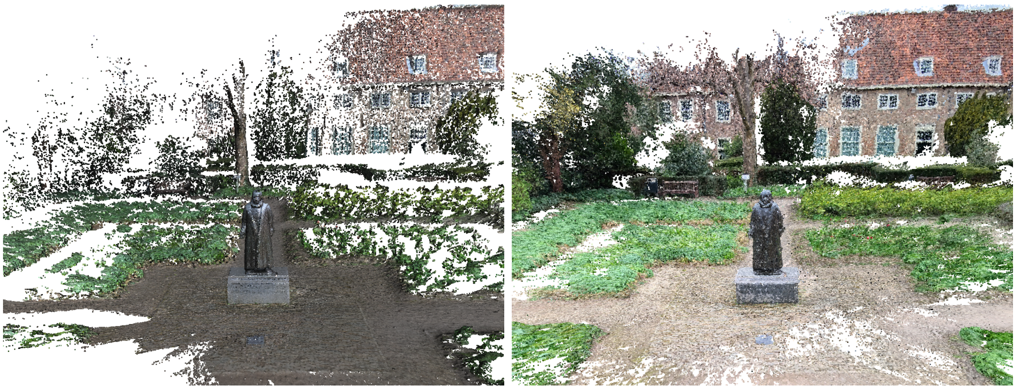

Figure 1.2: Top: Traditional 3D reconstruction method [10]
Bottom: our learning-based method
(Image credit: TU Delft students)

2 Stereo Reconstruction

Here, we will describe the depth estimation in two-view and multi-view settings where the pose information is known. We describe both traditional methods and learning-based methods.

2.1 Classical Two View Stereo Reconstruction [Optional]

2.1.1 Epipolar Geometry and Stereo Triangulation

Triangulation methods

Monocular vision has a scale ambiguity issue which makes it impossible to triangulate the scene with the correct scale. In a simple explanation, if the distance of the scene from the camera and geometry of the scene were scaled by some positive factor k, independently from the value of the k image plane will always have the same projection of the scene.

$$\begin{aligned} (X,Y,Z)^T &\longmapsto (fX/Z+o_x, fY/Z+o_y)^T \\ (kX,kY,kZ)^T &\longmapsto (fkX/kZ+o_x, fkY/kZ+o_y)^T = (fX/Z+o_x, fY/Z+o_y)^T \end{aligned} \tag{2.1}$$

Without any prior information, it is also impossible to perceive scene geometry from a single RGB image. The most popular way of constructing and perceiving scene geometry is having a motion to have a different camera view as shown in Figure 2.1.

Even having multiple monocular views without knowing extrinsic calibration will not resolve scale ambiguity. Again relative pose between views and camera to scene distance were scaled by some positive factor k, independently from the value of the k image plane will always have the same projection of the scene. Figure 2.2 shows that the point will have the same projection independent from the scale factor of k.

This section will mainly cover, scene triangulation in two and multi-view settings with known relative pose transformations.

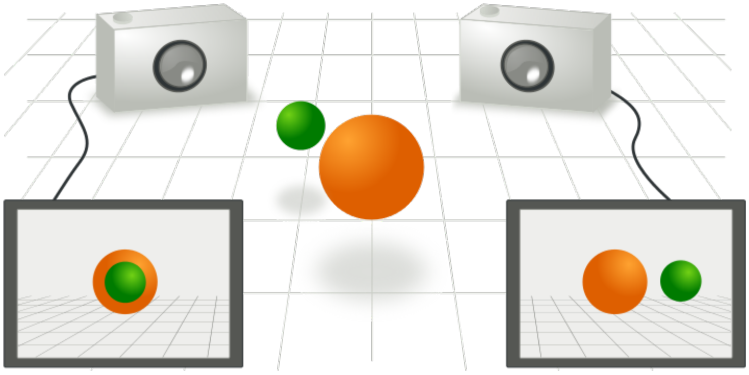

Figure 2.1: Left frame does not give much information if there is one or two spheres in the scene, seeing also right frame gives better understanding to viewer and lets viewer have perception of two spheres with different colors. (Image credit: Arne Nordmann)

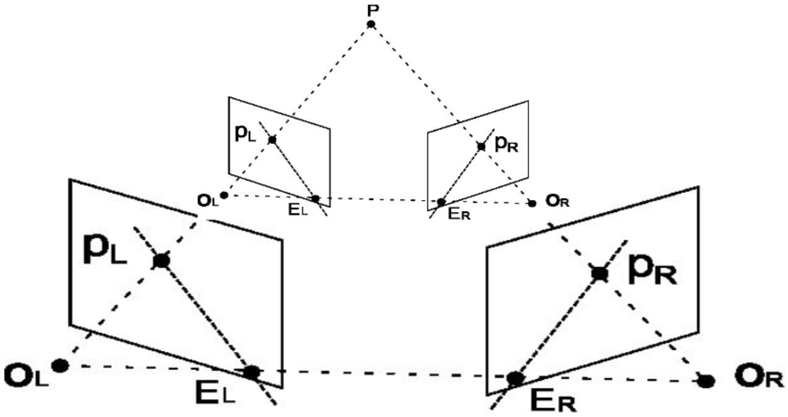

Figure 2.2: Scale ambiguity of the two view system without knowing the relative pose.

Two-view triangulation

Before diving into the two-view triangulation methods, this subsection will introduce the multiple view geometry basics and conventions. Let's assume $x_!$ and x_2 are the projection of 3D point X in homogeneous coordinates in two different frames. R and T are rotation and translation from the first frame to the second frame. λ_1 and λ_2 are distances from the camera centers to the 3D point X.

$$\begin{gathered}
\lambda_1 x_1 = X \quad and \quad \lambda_2 x_2 = RX + T \\
\lambda_2 x_2 = R(\lambda_1 x_1) + T \\
\hat{v}v = v \times v = 0 \quad \text{hat operator} \\
\lambda_2 \hat{T} x_2 = \hat{T} R(\lambda_1 x_1) + \hat{T} T = \hat{T} R(\lambda_1 x_1) \\
\lambda_2 x_2^T (\hat{T} x_2) = \lambda_1 x_2^T \hat{T} R x_1 \\
x2 \perp \hat{T} x_2 \\
x_2^T \hat{T} R x_1 = 0 \quad \text{epipolar constraint} \\
E = \hat{T} R \quad \text{essential matrix}
\end{gathered} \tag{2.2}$$

x'_i being image coordinate of x_i and the K being intrinsic matrix the equation can be formulated more generic for uncalibrated views.

$$\begin{gathered}
x_2'^T K^{-T} \hat{T} R K^{-1} x_1' = 0 \\
F = K^{-T} \hat{T} R K^{-1} \quad \text{fundamental matrix} \\
x_2'^T F x_1' = 0 \\
E = K^T F K \quad \text{relation between essential and fundamental matrix}
\end{gathered} \tag{2.3}$$

Because of the sensor noise and discretization step in image formation, there is noise in pixel coordinates of the 3D point projections. So the extensions of corresponding points in image planes usually do not intersect in 3D. This noise should be considered for getting accurate triangulation of corresponding points. There are multiple ways of doing two-view triangulation. Two of those methods will be covered here.

Midpoint method

A midpoint triangulation method is a simple approach for two-view triangulation. As shown in figure 2.3, the idea is finding the closest distance between the bearing vectors which are rays extensions from the camera intersecting the image plane at corresponding points. Q_1 and Q_2 are the points on these rays where the rays are at the

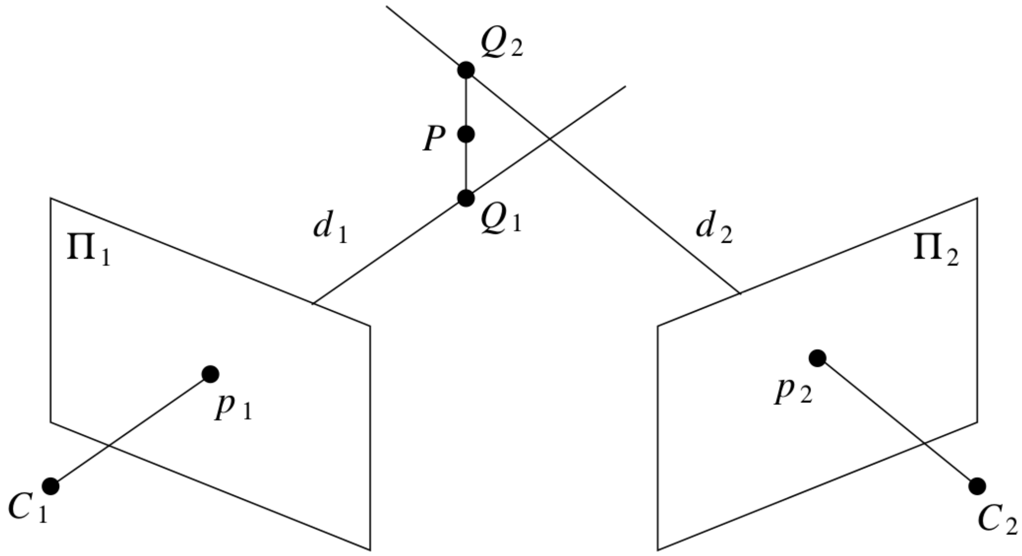

Figure 2.3: Midpoint point for two-view triangulation.

closest point to each other. The line passing through Q_1Q_2 should be perpendicular to these bearing vectors for Q_1Q_2 being the closest distance between these rays. The midpoint of the Q_1Q_2 is accepted as a valid 3D triangulation of the corresponding points. λ_i is being the scalar distance from the camera center to the 3D point Q_i, R and T are being relative pose from the second camera frame to the first camera, the approach mathematically can be formulated as below:

$$
\begin{gathered}
Q_1 = \lambda_1 d_1 \quad Q_2 = \lambda_2 R d_2 + T \quad (C_1 \text{ is chosen to be origin for simplicity}) \\
(Q_1 - Q_2)^T d_1 = 0 \quad (Q_1 - Q_2)^T R d_2 = 0 \text{ (dot product of perpendicular lines)} \\
\lambda_1 d_1^T d_1 - \lambda_2 R d_2^T d_1 = T^T d_1 \\
\lambda_1 d_1^T R d_2 - \lambda_2 R d_2^T R d_2 = T^T R d_2 \\
\begin{pmatrix} d_1^T d_1 & -(Rd_2)^T d_1 \\ (Rd_2)^T d_1 & (Rd_2)^T (Rd_2) \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} T^T d_1 \\ T^T (Rd_2) \end{pmatrix} \quad \text{(Ax = b) form equation} \\
\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = A^{-1} b \\
P = (Q_1 + Q_2)/2 = (\lambda_1 d_1 + \lambda_2 R d_2 + T)/2
\end{gathered}
\tag{2.4}
$$

Linear triangulation

This method is based on the fact that in the ideal case back-projected rays and rays from the camera center to the correspondence on the image plane should be aligned. The cross-product of these two vectors should be equal to zero in the ideal case. Using this knowledge problem converted to a set of linear equations that can be solved by SVD. Let x and y be correspondences, P, and P are respective perspective projection matrices of the camera, and λ_x and λ_y scalar values.

$$
\begin{gathered}
x = \begin{pmatrix} u_x \\ v_x \\ 1 \end{pmatrix} \quad y = \begin{pmatrix} u_y \\ v_y \\ 1 \end{pmatrix} \\
\lambda_x x = PX \quad \lambda_x y = QX \\
x \times PX = 0 \quad y \times QX = 0 \\
\begin{pmatrix} u_x \\ v_x \\ 1 \end{pmatrix} \times \begin{pmatrix} p_1^T \\ p_2^T \\ p_3^T \end{pmatrix} X = 0 \quad \begin{pmatrix} u_y \\ v_y \\ 1 \end{pmatrix} \times \begin{pmatrix} q_1^T \\ q_2^T \\ q_3^T \end{pmatrix} X = 0 \\
\begin{pmatrix} v_x p_3^T - p_2^T \\ p_1^T - u_x p_3^T \\ u_x p_2^T - v_x p_1^T \end{pmatrix} X = 0 \quad \begin{pmatrix} v_y q_3^T - q_2^T \\ q_1^T - u_y q_3^T \\ u_y q_2^T - v_y q_1^T \end{pmatrix} X = 0 \\
\begin{pmatrix} v_x p_3^T - p_2^T \\ p_1^T - u_x p_3^T \\ v_y q_3^T - q_2^T \\ q_1^T - u_y q_3^T \end{pmatrix} X = 0 \\
AX = 0
\end{gathered}
\tag{2.5}
$$

Solutions for X can be easily calculated using the Singular Value Decomposition.

Multi-view triangulation

For multi-view triangulation, one solution can be to find such an X in 3D space which has a minimum sum of square distance with the 3D points lying on the bearing vector. Analytical solution for X can be found by taking the derivative of the loss function with respect to the 3D point and finding the 3D point where is the derivative of loss function equals zero. Assuming the C_i is camera center of i^{th} camera, P_i is the point on i^{th} bearing vector, λ_i scalar distance between C_i and P_i, and X is being optimal 3D point as a triangulation result , the triangulation result can be formulated as below.

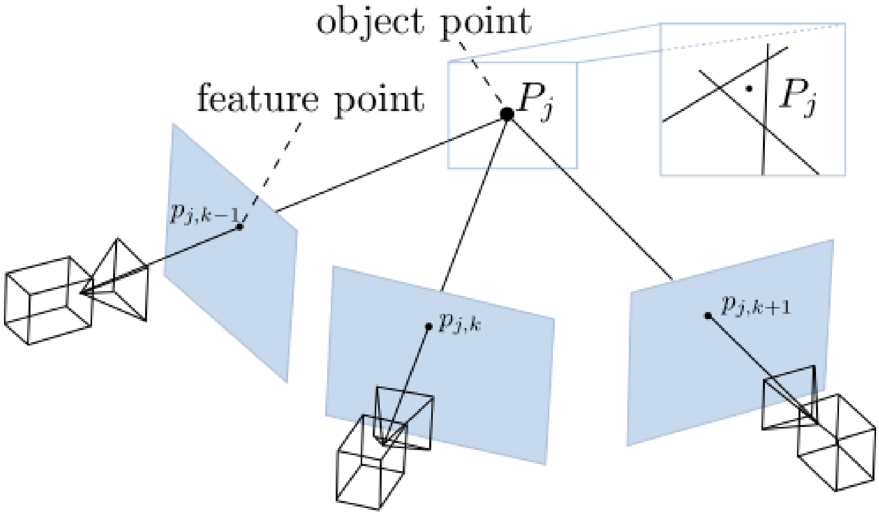

Figure 2.4: Multi view triangulation method. [9]

$$
\begin{gathered}
P_i = C_i + \lambda_i d_i \\
\lambda_i d_i \sim X - C_i \quad \text{At ideal case with no noise} \\
\lambda_i = \lambda_i d_i^T d_i \sim d_i^T (X - C_i) \\
P_i = C_i + \lambda_i d_i \sim C_i + d_i d_i^T (X - C_i) \\
r = X - C_i - d_i d_i^T (X - C_i) = (I - d_i d_i^T)(X - C_i) \\
\mathcal{L} = \sum_{i=1}^{N} r^2 = \sum_{i=1}^{N} ((I - d_i d_i^T)(X - C_i))^2 \\
\arg\min_x \mathcal{L} \Rightarrow \frac{\partial \mathcal{L}}{\partial X} = 0 \\
\frac{\partial \mathcal{L}}{\partial X} = 2 \sum_{i=1}^{N} (I - d_i d_i^T)^2 (X - C_i) = 0 \\
A_i = (I - d_i d_i^T) \Rightarrow \frac{\partial \mathcal{L}}{\partial X} = \sum_{i=1}^{N} A_i^T A_i (X - C_i) = 0 \\
X = (\sum_{i=1}^{N} A_i^T A_i)^{-1} \sum_{i=1}^{N} A_i^T A_i C_i
\end{gathered}
\tag{2.6}
$$

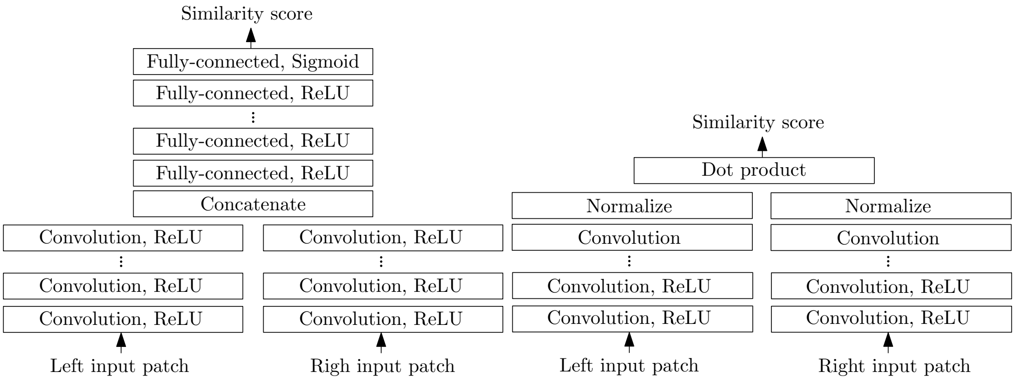

Figure 2.5: Simaese networks for stereo matching.[15]

2.2 Learning based Two View Stereo Reconstruction

Zbontar et al. [15] have initially shown that depth information can be extracted from rectified image pairs by learning a similarity measure on relevant image patches. They train their CNN-based siamese network as a binary classification network with similar and irrelevant pairs of patches.

Kendall et al. [7] proposed the network where they use 2D CNN with shared weights to retrieve rectified image pair features. In their work, they later used these feature maps to calculate a matching score-based cost volume, and as the last step, they use a 3D CNN-based autoencoder to regularize this volume.

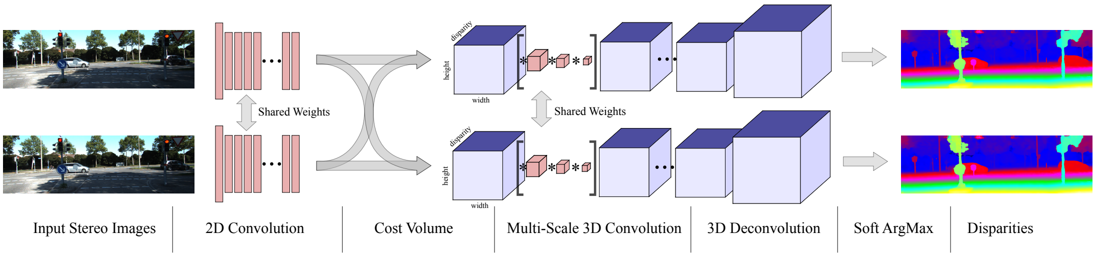

Figure 2.6: GC-Net - deep stereo regression architecture.[7]

2.3 PMVS

Patch-Based Multi-View Stereo [2] has proven being quite effective in practice. After an initial feature matching step aimed at constructing a sparse set of photoconsistent patches, in the sense of the previous section—that is, patches whose projections in the images where they are visible have similar brightness or color patterns—it divides the input images into small square cells a few pixels across, and attempts to reconstruct a patch in each one of them, using the cell connectivity to propose new patches, and visibility constraints to filter out incorrect ones.

We assume throughout that n cameras with known intrinsic and extrinsic parameters observe a static scene, and respectively denote by O_i and I_i $(i = 1, ..., n)$ the optical centers of these cameras and the images they have recorded of the scene. The main elements of the PMVS model of multi-view stereo fusion and scene reconstruction are small rectangular patches, intended to be tangent to the observed surfaces, and a few of these patches' key properties—namely, their geometry, which images they are visible in and whether they are photoconsistent with those, and some notion of connectivity inherited from image topology.

1. **Matching** Use feature matching to construct an initial set of patches, and optimize their parameters to make them maximally photoconsistent.
2. Repeat 3 times:
   a) **Expansion** Iteratively construct new patches in empty spots near existing ones, using image connectivity and depth extrapolation to propose candidates, and optimizing their parameters as before to make them maximally photoconsistent.
   b) **Filtering** Use again the image connectivity to remove patches identified as outliers because their depth is not consistent with a sufficient number of other nearby patches.

*You should read **section 2 (Key Elements of the Proposed Approach)** and **section 3 (Algorithm)** from the paper.*

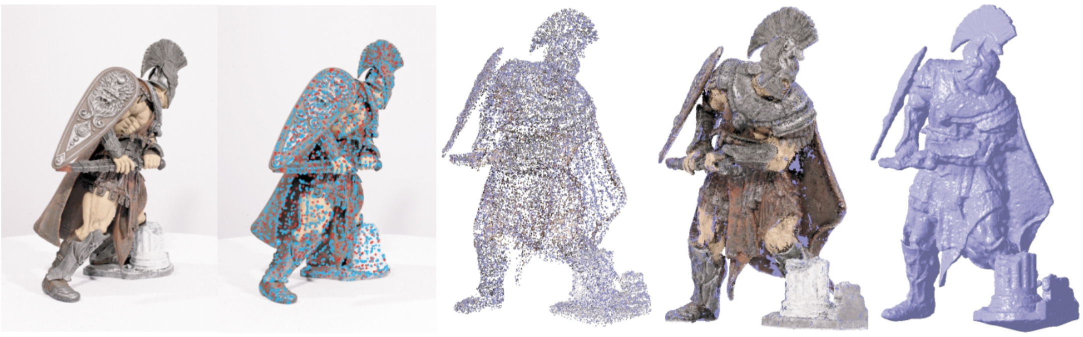

Figure 2.7: PMVS - overall approach. From left to right: a sample input image; detected features; reconstructed patches after the initial matching;final patches after expansion and filtering; polygonal surface extracted from reconstructed patches.[2]

2.4 MVSNet

State-of-the-art learning-based MVS approaches adapt the photogrammetry-based MVS algorithms by implementing them as a set of differentiable operations defined in the feature space. MVSNet [13] introduced good quality 3D reconstruction by regularizing the cost volume that was computed using differentiable homography on feature maps of the reference and source images.

2.4.1 Differentiable homography

We know that any plane in 3 dimensional space can be parametrized with normal n^T and distance d. For plane π in the figure we can say that point P_i lies on the plane if and only if $n^T P_i + d = 0$.

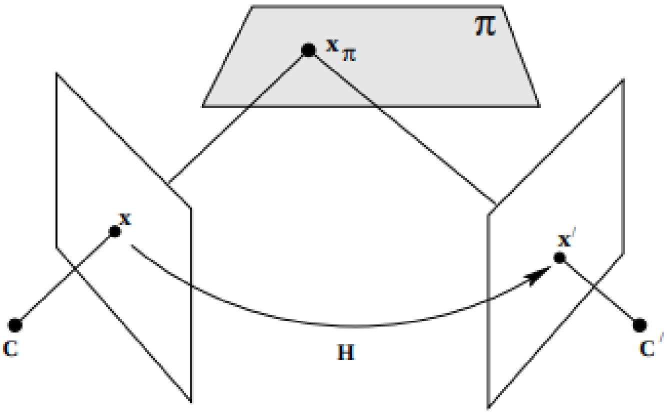

Figure 2.8: Two view homography [5] [4] [12]

Derivation with relative pose

$$\begin{gathered}
p_a = \frac{1}{z_a} K_a H_{ab} z_b K_b^{-1} p_b = \frac{z_b}{z_a} K_a H_{ab} K_b^{-1} p_b \\
\text{"R" and "t" are relative pose of a with respect to b} \\
H_{ab} P_b = R * P_b + t \\
\text{Plane constraint } n^T P_b + d = 0. \\
H_{ab} P_b = R P_b + t \frac{-n^T P_b}{d} \quad \text{since } -n^T P_b / d = 1 \\
H_{ab} = R - \frac{n^T t}{d}
\end{gathered} \tag{2.7}$$

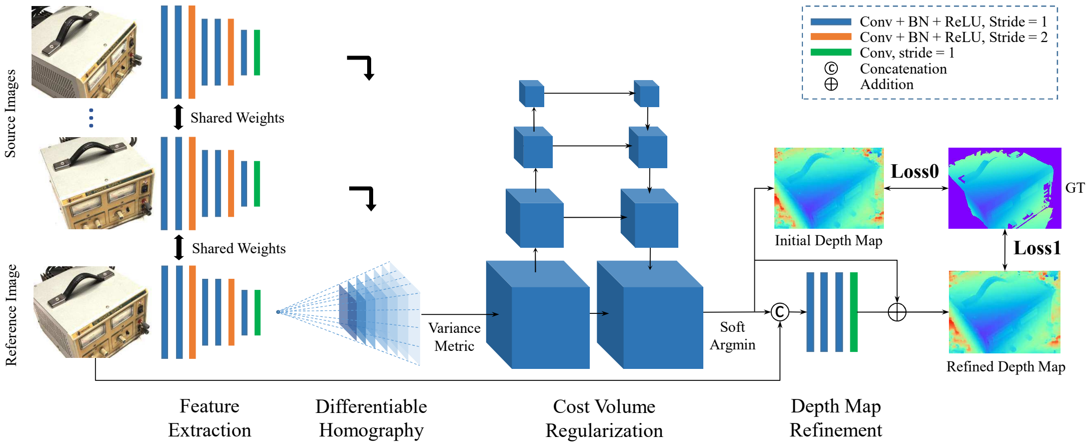

Figure 2.9: MVSNet architecture.[13]

Derivation with absolute pose

$$\begin{aligned}
H_{ab} &= R - \frac{n^T t}{d} \quad \text{remember relative case} \\
P_b &= R_b P_w + t_b \quad \text{from world to camera b} \\
P_a &= R_a P_w + t_a \quad \text{from world to camera a} \\
P_w &= R_b^T P_b - R_b^T t_b \\
P_a &= R_a P_w + t_a = R_a (R_b^T P_b - R_b^T t_b) + t_a \\
P_a &= R_a P_w + t_a = R_a R_b^T P_b + t_a - R_a R_b^T t_b \\
R_{a \leftarrow b} &= R_a R_b^T \quad t_{a \leftarrow b} = t_a - R_a R_b^T t_b \\
H_{ab} &= R_{a \leftarrow b} - \frac{n^T t_{a \leftarrow b}}{d} \quad \text{remember relative case} \\
H_{ab} &= R_a R_b^T - \frac{n^T (t_a - R_a R_b^T t_b)}{d}
\end{aligned} \tag{2.8}$$

2.4.2 MVSNet overall architecture

MVSNet at first extract the deep features of the N (number of views) input images for dense matching. It applies convolutional filters to extract the feature towers scale.Each convolutional layer is followed by a batch-normalization (BN) layer and a rectified linear unit (ReLU) the last layer.

Using features and the camera parameters, then we build cost volume regularization. We use differentiable homography for building this cost volumes.

The raw cost volume computed from image features are regularized later. Multi-scale 3DCNN have been used for cost volume regularization.This regularization step is designed for refining the above cost volume to generate a probability volume for depth inference.

Depth that was regressed from probability volume is further refined using the 2DCNN network.

*You should read **section 3 (3. MVSNET)** from the paper.*

Bibliography

[1] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, pages 1–16, 2016.

[2] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE TPAMI*, 32(8):1362–1376, 2010.

[3] S. Galliani, K. Lasinger, and K. Schindler. Massively parallel multiview stereopsis by surface normal diffusion. June 2015.

[4] D. Gallup, J. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. doi: 10.1109/CVPR.2007.383245.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[6] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

[7] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017.

[8] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017.

[9] P. Moulon, P. Monasse, R. Perrot, and R. Marlet. OpenMVG: Open multiple view geometry. In *International Workshop on Reproducible Research in Pattern Recognition*, pages 60–74. Springer, 2016.

[10] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[11] F. Wang, S. Galliani, C. Vogel, P. Speciale, and M. Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo, 2021.

[12] Wikipedia. Homography (computer vision). Wikipedia, 2013.

[13] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. Mvsnet: Depth inference for unstructured multi-view stereo. *European Conference on Computer Vision (ECCV)*, 2018.

[14] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, and L. Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[15] J. Zbontar, Y. LeCun, et al. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.*, 17(1):2287–2318, 2016.