Accurate, Dense, and Robust Multi-View Stereopsis

Yasutaka Furukawa[1]
Department of Computer Science
and Beckman Institute
University of Illinois at Urbana-Champaign, USA[1]

Jean Ponce[1,2]
Willow Team–ENS/INRIA/ENPC
Département d'Informatique
Ecole Normale Supérieure, Paris, France[2]

Abstract: *This paper proposes a novel algorithm for calibrated multi-view stereopsis that outputs a (quasi) dense set of rectangular patches covering the surfaces visible in the input images. This algorithm does not require any initialization in the form of a bounding volume, and it detects and discards automatically outliers and obstacles. It does not perform any smoothing across nearby features, yet is currently the top performer in terms of both coverage and accuracy for four of the six benchmark datasets presented in [20]. The keys to its performance are effective techniques for enforcing local photometric consistency and global visibility constraints. Stereopsis is implemented as a* match, expand, and filter *procedure, starting from a sparse set of matched keypoints, and repeatedly expanding these to nearby pixel correspondences before using visibility constraints to filter away false matches. A simple but effective method for turning the resulting patch model into a mesh appropriate for image-based modeling is also presented. The proposed approach is demonstrated on various datasets including objects with fine surface details, deep concavities, and thin structures, outdoor scenes observed from a restricted set of viewpoints, and "crowded" scenes where moving obstacles appear in different places in multiple images of a static structure of interest.*

1. Introduction

As in the binocular case, although most early work in multi-view stereopsis (e.g., [12, 15, 19]) tended to match and reconstruct all scene points independently, recent approaches typically cast this problem as a variational one, where the objective is to find the surface minimizing a global photometric discrepancy functional, regularized by explicit smoothness constraints [1, 8, 17, 18, 22, 23] (a geometric consistency terms is sometimes added as well [3, 4, 7, 9]). Competing approaches mostly differ in the type of optimization techniques that they use, ranging from local methods such as gradient descent [3, 4, 7], level sets [1, 9, 18], or expectation maximization [21], to global ones such as graph cuts [3, 8, 17, 22, 23]. The variational approach has led to impressive progress, and several of the methods recently surveyed by Seitz et al. [20] achieve a relative accuracy better than 1/200 (1mm for a 20cm wide object) from a set of low-resolution (640×480) images. However, it typically requires determining a bounding volume (valid depth range, bounding box, or visual hull) prior to initiating the optimization process, which may not be feasible for outdoor scenes and/or cluttered images. [1] We propose instead a simple and efficient algorithm for calibrated multi-view stereopsis that does not require any initialization, is capable of detecting and discarding outliers and obstacles, and outputs a (quasi) dense collection of small oriented rectangular patches [6, 13], obtained from pixel-level correspondences and tightly covering the observed surfaces except in small textureless or occluded regions. It does not perform any smoothing across nearby features, yet is currently the top performer in terms of both coverage and accuracy for four of the six benchmark datasets provided in [20]. The keys to its performance are effective techniques for enforcing local photometric consistency and global visibility constraints. Stereopsis is implemented as a *match, expand, and filter* procedure, starting from a sparse set of matched keypoints, and repeatedly expanding these to nearby pixel correspondences before using visibility constraints to filter away false matches. A simple but effective method for turning the resulting patch model into a mesh suitable for image-based modeling is also presented. The proposed approach is applied to three classes of datasets:

- *objects*, where a single, compact object is usually fully visible in a set of uncluttered images taken from all around it, and it is relatively straightforward to extract the apparent contours of the object and compute its visual hull;
- *scenes*, where the target object(s) may be partially occluded and/or embedded in clutter, and the range of viewpoints may be severely limited, preventing the computation of effective bounding volumes (typical examples are outdoor scenes with buildings or walls); and

[1] In addition, variational approaches typically involve massive optimization tasks with tens of thousands of coupled variables, potentially limiting the resolution of the corresponding reconstructions (see, however, [18] for a fast GPU implementation). We will revisit tradeoffs between computational efficiency and reconstruction accuracy in Sect. 5.

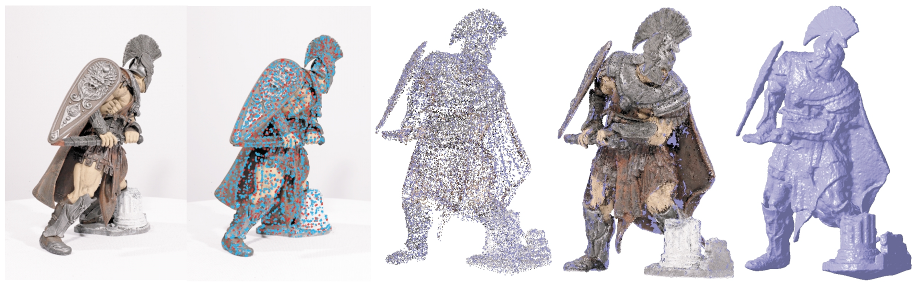


Figure 1. Overall approach. From left to right: a sample input image; detected features; reconstructed patches after the initial matching; final patches after expansion and filtering; polygonal surface extracted from reconstructed patches.

• *crowded scenes*, where moving obstacles appear in different places in multiple images of a static structure of interest (e.g., people passing in front of a building).

Techniques such as space carving [12, 15, 19] and variational methods based on gradient descent [3, 4, 7], level sets [1, 9, 18], or graph cuts [3, 8, 17, 22, 23] typically require an initial bounding volume and/or a wide range of viewpoints. Object datasets are the ideal input for these algorithms, but methods using multiple depth maps [5, 21] or small, independent surface elements [6, 13] are better suited to the more challenging scene datasets. Crowded scenes are even more difficult. The method proposed in [21] uses expectation maximization and multiple depth maps to reconstruct a crowded scene despite the presence of occluders, but it is limited to a small number of images (typically three). As shown by qualitative and quantitative experiments in the rest of this paper, our algorithm effectively handles all three types of data, and, in particular, outputs accurate object and scene models with fine surface detail despite low-texture regions, large concavities, and/or thin, high-curvature parts. As noted earlier, it implements multi-view stereopsis as a simple *match, expand, and filter* procedure (Fig. 1): (1) *Matching*: features found by Harris and Difference-of-Gaussians operators are matched across multiple pictures, yielding a sparse set of patches associated with salient image regions. Given these initial matches, the following two steps are repeated n times ($n = 3$ in all our experiments): (2) *Expansion:* a technique similar to [16, 2, 11, 13] is used to spread the initial matches to nearby pixels and obtain a dense set of patches. (3) *Filtering:* visibility constraints are used to eliminate incorrect matches lying either in front or behind the observed surface. This approach is similar to the method proposed by Lhuillier and Quan [13], but their expansion procedure is greedy, while our algorithm iterates between expansion and filtering steps, which allows us to process complicated surfaces. Furthermore, outliers cannot be handled in their method. These differences are also true with the approach by Kushal and Ponce [11] in comparison to ours. In addition, only a pair of images can be handled at once in [11], while our method can process arbitrary number of images uniformly.

2. Key Elements of the Proposed Approach

Before detailing our algorithm in Sect. 3, we define here the patches that will make up our reconstructions, as well as the data structures used throughout to represent the input images. We also introduce two other fundamental building blocks of our approach, namely, the methods used to accurately reconstruct a patch once the corresponding image fragments have been matched, and determine its visibility.

2.1. Patch Models

A patch p is a rectangle with center $c(p)$ and unit normal vector $n(p)$ oriented toward the cameras observing it (Fig. 2). We associate with p a reference image $R(p)$, chosen so that its retinal plane is close to parallel to p with little distortion. In turn, $R(p)$ determines the orientation and extent of the rectangle p in the plane orthogonal to $n(p)$, so the projection of one of its edges into $R(p)$ is parallel to the image rows, and the smallest axis-aligned square containing its image covers a $\mu \times \mu$pixel^2 area (we use values of 5 or 7 for μ in all of our experiments). Two sets of pictures are also attached to each patch p: the images $S(p)$ where p *should* be visible (despite self-occlusion), but may in practice not be recognizable (due to highlights, motion blur, etc.), or hidden by moving obstacles, and the images $T(p)$ where it is *truly* found ($R(p)$ is of course an element of $T(p)$). We enforce the following two constraints on the model: First, we enforce local *photometric consistency* by requiring that the projected textures of every patch p be consistent in at least γ images (in other words $|T(p)| \geq \gamma$, with $\gamma = 3$ in all but three of our experiments, where γ is set to 2). Second, we enforce global *visibility consistency* by requiring that no patch p be occluded by any other patch in any image in $S(p)$. [2]

2.2. Image Models

We associate with each image I a regular grid of $\beta_1 \times \beta_1$pixel^2 cells $C(i, j)$, and attempt to reconstruct at least one

[2]A patch p *may* be occluded in one or several of the images in $S(p)$ by moving obstacles, but these are *not* reconstructed by our algorithm and thus do not generate occluding patches.

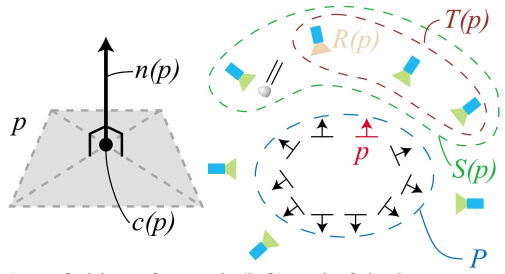


Figure 2. Definition of a patch (left) and of the images associated with it (right). See text for the details.

patch in every cell (we use values of 1 or 2 for β_1 in all our experiments). The cell $C(i,j)$ keeps track of two different sets $Q_t(i,j)$ and $Q_f(i,j)$ of reconstructed patches potentially visible in $C(i,j)$: A patch p is stored in $Q_t(i,j)$ if $I \in T(p)$, and in $Q_f(i,j)$ if $I \in S(p) \setminus T(p)$. We also associate with $C(i,j)$ the depth of the center of the patch in $Q_t(i,j)$ closest to the optical center of the corresponding camera. This amounts to attaching a depth map to I, which will prove useful in the visibility calculations of Sect. 2.4.

2.3. Enforcing Photometric Consistency

Given a patch p, we use the normalized cross correlation (NCC) $N(p,I,J)$ of its projections into the images I and J to measure their photometric consistency. Concretely, a $\mu \times \mu$ grid is overlaid on p and projected into the two images, the correlated values being obtained through bilinear interpolation. Given a patch p, its reference image $R(p)$, and the set of images $T(p)$ where it is truly visible, we can now estimate its position $c(p)$ and its surface normal $n(p)$ by maximizing the average NCC score

$$\bar{N}(p) = \frac{1}{|T(p)|-1} \sum_{I \in T(p), I \neq R(p)} N(p,R(p),I) \tag{1}$$

with respect to these unknowns. To simplify computations, we constrain $c(p)$ to lie on the ray joining the optical center of the reference camera to the corresponding image point, reducing the number of degrees of freedom of this optimization problem to three—depth along the ray plus yaw and pitch angles for $n(p)$, and use a conjugate gradient method [14] to find the optimal parameters. Simple methods for computing reasonable initial guesses for $c(p)$ and $n(p)$ are given in Sects. 3.1 and 3.2.

2.4. Enforcing Visibility Consistency

The visibility of each patch p is determined by the images $S(p)$ and $T(p)$ where it is (potentially or truly) observed. We use two slightly different methods for constructing $S(p)$ and $T(p)$ depending on the stage of our reconstruction algorithm. In the matching phase (Sect. 3.1), patches are reconstructed from sparse feature matches, and we have to rely on photometric consistency constraints to determine (or rather obtain an initial guess for) visibility. Concretely, we initialize both sets of images as those for which the NCC score exceeds some threshold: $S(p) = T(p) = \{I | N(p,R(p),I) > \alpha_0\}$. On the other hand, in the expansion phase of our algorithm (Sect. 3.2), patches are by construction dense enough to associate depth maps with all images, and $S(p)$ is constructed for each patch by thresholding these depth maps—that is, $S(p) = \{I | d_I(p) \leq d_I(i,j) + \rho_1\}$, where $d_I(p)$ is the depth of the center of p along the corresponding ray of image I, and $d_I(i,j)$ is the depth recorded in the cell $C(i,j)$ associated with image I and patch p. The value of ρ_1 is determined automatically as the distance at the depth of $c(p)$ corresponding to an image displacement of β_1 pixels in $R(p)$. Once $S(p)$ has been estimated, photometric consistency is used to determine the images where p is truly observed as $T(p) = \{I \in S(p) | N(p,R(p),I) > \alpha_1\}$. This process may fail when the reference image $R(p)$ is itself an outlier, but, as explained in the next section, our algorithm is designed to handle this problem. Iterating its matching and filtering steps also helps improve the reliability and consistency of the visibility information.

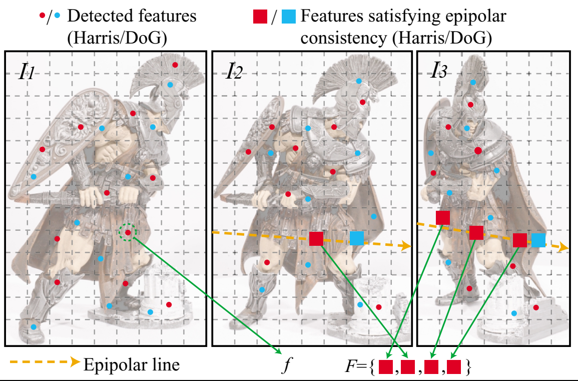
3. Algorithm

3.1. Matching

As the first step of our algorithm, we detect corner and blob features in each image using the Harris and Difference-of-Gaussian (DoG) operators. [3] To ensure uniform coverage, we lay over each image a coarse regular grid of $\beta_2 \times \beta_2$pixel^2 cells, and return as corners and blobs for each cell the η local maxima of the two operators with strongest responses (we use $\beta_2 = 32$ and $\eta = 4$ in all our experiments). After these features have been found in each image, they are matched across multiple pictures to reconstruct a sparse set of patches, which are then stored in the grid of cells $C(i,j)$ overlaid on each image (Fig. 3): Consider an image I and denote by O the optical center of the corresponding camera. For each feature f detected in I, we collect in the other images the set F of features f' of the same type (Harris or DoG) that lie within $\iota = 2$pixels from the corresponding epipolar lines, and triangulate the 3D points associated with the pairs (f,f'). We then consider these points in order of increasing distance from O as potential patch centers, [4] and return the first patch "photoconsistent" in at least γ images (Fig. 3, top). More concretely, for each

[3]Briefly, let us denote by G_σ a 2D Gaussian with standard deviation σ. The response of the Harris filter at some image point is defined as $H = \det(M) - \lambda \text{trace}^2(M)$, where $M = G_{\sigma_0} * (\nabla I \nabla I^T)$, and ∇I is computed by convolving the image I with the partial derivatives of the Gaussian G_{σ_1}. The response of the DoG filter is $D = |(G_{\sigma_2} - G_{\sqrt{2}\sigma_2}) * I|$. We use $\sigma_0 = \sigma_1 = \sigma_2 = 1$pixel and $\lambda = 0.06$ in all of our experiments.

[4]Empirically, this heuristic has proven to be effective in selecting mostly correct matches at a modest computational expense.



●/● Detected features (Harris/DoG)
🟥/🟦 Features satisfying epipolar consistency (Harris/DoG)
I_1
I_2
I_3
Epipolar line
f
F={🟥,🟥,🟥,🟥}

Input: Features detected in each image.
Output: Initial sparse set of patches P.

```
Cover each image with a grid of $\beta_1 \times \beta_1$pixel$^2$ cells;
$P \leftarrow \phi$;
For each image $I$ with optical center $O$
  For each feature $f$ detected in $I$ and lying in an empty cell
    $F \leftarrow$ {Features satisfying the epipolar consistency};
    Sort $F$ in an increasing order of distance from $O$;
    For each feature $f' \in F$
      $R(p) \leftarrow I$; $T(p) \leftarrow \{J | N(p, R(p), J) \geq \alpha_0\}$;
      $c(p) \leftarrow$ 3D point triangulated from $f$ and $f'$;
      $n(p) \leftarrow$ Direction of optical ray from $c(p)$ to $O$;
      $n(p), c(p) \leftarrow$ argmax $\bar{N}(p)$;
      $S(p) \leftarrow \{J | N(p, R(p), J) \geq \alpha_0\}$;
      $T(p) \leftarrow \{J | N(p, R(p), J) \geq \alpha_1\}$;
      If $|T(p)| \geq \gamma$
        register $p$ to the corresponding cells in $S(p)$;
        exit innermost For loop, and add $p$ to $P$.
```

Figure 3. Feature matching algorithm. Top: An example showing the features $f' \in F$ satisfying the epipolar constraint in images I_2 and I_3 as they are matched to feature f in image I_1 (this is an illustration only, not showing actual detected features). Bottom: The matching algorithm. The values used for α_0 and α_1 in all our experiments are 0.4 and 0.7 respectively.

feature f', we construct the potential surface patch p by triangulating f and f' to obtain an estimate of $c(p)$, assign to $n(p)$ the direction of the optical ray joining this point to O, and set $R(p) = I$. After initializing $T(p)$ by using photometric consistency as in Sect. 2.4, we use the optimization process described in Sect. 2.3 to refine the parameters of $c(p)$ and $n(p)$, then initialize $S(p)$ and recompute $T(p)$. Finally, if p satisfies the constraint $|T(p)| \geq \gamma$, we compute its projections in all images in $S(p)$, register it to the corresponding cells, and add it to P (Fig. 3, bottom). Note that since the purpose of this step is only to reconstruct an initial, sparse set of patches, features lying in non-empty cells are skipped for efficiency. Also note that the patch generation process may fail if the reference image $R(p)$ is an outlier, for example when f correspond to a highlight. This does not prevent, however, the reconstruction of the corresponding surface patch from another image. The second part of our algorithm iterates (three times in all our experiments) between an expansion step to obtain dense patches and a filtering step to remove erroneous matches and enforce visibility consistency, as detailed in the next two sections.

3.2. Expansion

At this stage, we iteratively add new neighbors to existing patches until they cover the surfaces visible in the scene. Intuitively, two patches p and p' are considered to be neighbors when they are stored in adjacent cells $C(i, j)$ and $C(i', j')$ of the same image I in $S(p)$, and their tangent planes are close to each other. We only attempt to create new neighbors when necessary—that is, when $Q_t(i', j')$ is empty,[5] and none of the elements of $Q_f(i', j')$ is *n-adjacent* to p, where two patches p and p' are said to be n-adjacent when $|(c(p) - c(p')) \cdot n(p)| + |(c(p) - c(p')) \cdot n(p')| < 2\rho_2$. Similar to ρ_1, ρ_2 is determined automatically as the distance at the depth of the mid-point of $c(p)$ and $c(p')$ corresponding to an image displacement of β_1 pixels in $R(p)$. When these two conditions are verified, we initialize the patch p' by assigning to $R(p')$, $T(p')$, and $n(p')$ the corresponding values for p, and assigning to $c(p')$ the point where the viewing ray passing through the center of $C(i', j')$ intersects the plane containing the patch p. Next, $c(p')$ and $n(p')$ are refined by the optimization procedure discussed in Sect. 2.3, and $S(p')$ is initialized from the depth maps as explained in Sect. 2.4. Since some matches (and thus the corresponding depth map information) may be incorrect at this point, the elements of $T(p')$ are added to $S(p')$ to avoid missing any image where p' may be visible. Finally, after updating $T(p')$ using photometric constraints as in Sect. 2.4, we accept the patch p' if $|T(p')| \geq \gamma$ still holds, then register it to $Q_t(i', j')$ and $Q_f(i', j')$, and update the depth maps associated with images in $S(p')$. See Fig. 4 for the algorithm.

3.3. Filtering

Two filtering steps are applied to the reconstructed patches to further enforce visibility consistency and remove erroneous matches. The first filter focuses on removing patches that lie outside the real surface (Fig. 5, left): Consider a patch p_0 and denote by U the set of patches that it occludes. We remove p_0 as an outlier when $|T(p_0)|\bar{N}(p_0) < \sum_{p_j \in U} \bar{N}(p_j)$ (intuitively, when p_0 is an outlier, both $\bar{N}(p_0)$ and $|T(p_0)|$ are expected to be small, and p_0 is likely to be removed). The second filter focuses on outliers lying inside the actual surface (Fig. 5, right): We simply recompute $S(p_0)$ and $T(p_0)$ for each patch p_0 using the depth maps associated with the corresponding images (Sect. 2.4), and

[5]Intuitively, any patch p' in $Q_t(i', j')$ would either already be a neighbor of p, or be separated from it by a depth discontinuity, neither case warranting the addition of a new neighbor.

```
Input: Patches $P$ from the feature matching step.
Output: Expanded set of reconstructed patches.

Use $P$ to initialize, for each image, $Q_f$, $Q_t$, and its depth map.
While $P$ is not empty
  Pick and remove a patch $p$ from $P$;
  For each image $I \in T(p)$ and cell $C(i,j)$ that $p$ projects onto
    For each cell $C(i',j')$ adjacent to $C(i,j)$ such that $Q_t(i',j')$
    is empty and $p$ is not n-adjacent to any patch in $Q_f(i',j')$
      Create a new $p'$, copying $R(p'), T(p')$ and $n(p')$ from $p$;
      $c(p') \leftarrow$ Intersection of optical ray through
                    center of $C(i',j')$ with plane of $p$;
      $n(p'), c(p') \leftarrow \operatorname{argmax} \bar{N}(p')$;
      $S(p') \leftarrow$ {Visible images of $p'$ estimated by the
                   current depth maps } $\cup\, T(p')$;
      $T(p') \leftarrow \{J \in S(p') | N(p', R(p'), J) \geq \alpha_1\}$;
      If $|T(p') < \gamma|$, go back to For-loop;
      Add $p'$ to $P$;
      Update $Q_t, Q_f$ and depth maps for $S(p')$;
Return all the reconstructed patches stored in $Q_f$ and $Q_t$.
```

Figure 4. Patch expansion algorithm.

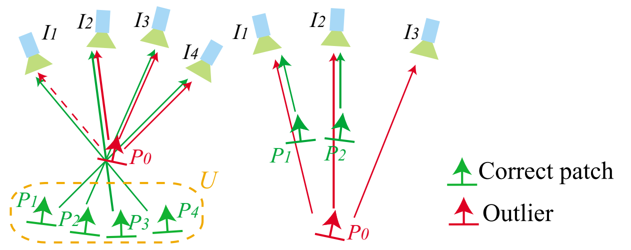


Figure 5. Outliers lying outside (left) or inside (right) the correct surface. Arrows are drawn between the patches p_i and the images I_j in $S(p_i)$, while solid arrows correspond to the case where $I_j \in T(p_i)$. U denotes a set of patches occluded by an outlier. See text for details.

remove p_0 when $|T(p_0)| < \gamma$. Note that the recomputed values of $S(p_0)$ and $T(p_0)$ may be different from those obtained in the expansion step since more patches have been computed after the reconstruction of p_0. Finally, we enforce a weak form of regularization as follows: For each patch p, we collect the patches lying in its own and adjacent cells in all images of $S(p)$. If the proportion of patches that are *n-adjacent* to p in this set is lower than $\varepsilon = 0.25$, p is removed as an outlier. The threshold α_1 is initialized with 0.7, and lowered by 0.2 after each expansion/filtering iteration.

4. Polygonal Surface Reconstruction

The reconstructed patches form an *oriented point*, or *surfel* model. Despite the growing popularity of this type of models in the computer graphics community [10], it remains desirable to turn our collection of patches into surface meshes for image-based modeling applications. The

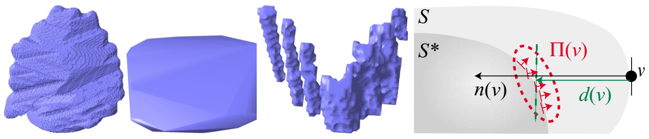


Figure 6. Polygonal surface reconstruction. Left: bounding volumes for the *dino* (visual hull), *steps* (convex hull), and *city-hall* (union of hemispheres) datasets featured in Figs. 7 ,9 and 10. Right: geometric elements driving the deformation process.

approach that we have adopted is a variant of the iterative deformation algorithm presented in [4], and consists of two phases. Briefly, after initializing a polygonal surface from a predetermined bounding volume, the convex hull of the reconstructed points, or a set of small hemispheres centered at these points and pointing away from the cameras, we repeatedly move each vertex v according to three forces (Fig. 6): a *smoothness* term for regularization; a *photometric consistency* term, which is based on the reconstructed patches in the first phase, but is computed solely from the mesh in the second phase; and, when accurate silhouettes are available, a *rim consistency* term pulling the rim of the deforming surface toward the corresponding visual cones.

Concretely, the smoothness term is $-\zeta_1 \Delta v + \zeta_2 \Delta^2 v$, where Δ denotes the (discrete) Laplacian operator relative to a local parameterization of the tangent plane in v ($\zeta_1 = 0.6$ and $\zeta_2 = 0.4$ are used in all our experiments). In the first phase, the photometric consistency term for each vertex v essentially drives the surface towards reconstructed patches and is given by $\nu(v)n(v)$, where $n(v)$ is the inward unit normal to S in v, $\nu(v) = \max(-\tau, \min(\tau, d(v)))$, and $d(v)$ is the signed distance between v and the true surface S^* along $n(v)$ (the parameter τ is used to bound the magnitude of the force, ensure stable deformation and avoid self-intersections; its value is fixed as 0.2 times the average edge length in S). In turn, $d(v)$ is estimated as follows: We collect the set $\Pi(v)$ of $\pi = 10$ patches p' with (outward) normals compatible with that of v (that is, $-n(p') \cdot n(v) > 0$, see Fig. 6) that lie closest to the line defined by v and $n(v)$, and compute $d(v)$ as the weighted average distance from v to the centers of the patches in $\Pi(v)$ along $n(v)$—that is, $d(v) = \sum_{p' \in \Pi(v)} w(p')[n(v) \cdot (c(p') - v)]$, where the weights $w(p')$ are Gaussian functions of the distance between $c(p')$ and the line, with standard deviation ρ_1 defined as before, and normalized to sum to 1. In the second phase, the photometric consistency term is computed for each vertex by using the patch optimization routine as follows. At each vertex v, we create a patch p by initializing $c(p)$ with v, $n(p)$ with a surface normal estimated at v on S, and a set of visible images $S(p)$ from a depth-map testing on the mesh S at v, then apply the patch optimization routine described in Sect. 2.3. Let $c^*(p)$ denote the value of $c(p)$ after the optimization, then $c^*(p) - c(p)$ is used as the photometric consistency term. In the first phase, we iterate until conver-

Table 1. Characteristics of the datasets used in our experiments. *roman* and *skull* datasets have been acquired in our lab, while other datasets have been kindly provided by S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski (*temple* and *dino*, see also [20]); C. Hernández Esteban, F. Schmitt and the Museum of Cherbourg (*polynesian*); S. Sullivan and Industrial Light and Magic (*face*, *face-2*, *body*, *steps*, and *wall*); and C. Strecha (*city-hall* and *brussels*).

Name	Images	Image Size	β_1	μ	γ
roman	48	1800×1200	1	5	3
temple	16	640×480	1	5	3
dino	16	640×480	1	7	3
skull	24	2000×2000	2	5	3
polynesian	36	1700×2100	2	5	3
face	4	1400×2200	1	7	2
face-2	13	1500×1400	1	5	3
body	4	1400×2200	1	7	2
steps	7	1500×1400	1	7	3
city-hall	5	3000×2000	2	7	3
wall	9	1500×1400	1	7	3
brussels	3	2000×1300	1	5	2

gence, remesh, increase the resolution of the surface, and repeat the process until the desired resolution is obtained (in particular, until image projections of edges of the mesh become approximately β_1 pixels in length, see [4] for details). The second phase is applied to the mesh only in its desired resolution as a final refinement.

5. Experiments and Discussion

We have implemented the proposed approach in C++, using the WNLIB [14] implementation of conjugate gradient in the patch optimization routine. The datasets used in our experiments are listed in Table 1, together with the number of input images, their approximate size and a choice of parameters for each data set. Note that all the parameters except for β_1, μ and γ have been fixed in our experiments.

We have first tested our algorithm on *object* datasets (Figs. 1 and 7) for which a segmentation mask is available in each image. A visual hull model is thus used to initialize the iterative deformation process for all these datasets, except for *face* and *body*, where a limited set of viewpoints is available, and the convex hull of the reconstructed patches is used instead. The segmentation mask is also used by our stereo algorithm, which simply ignores the background during feature detection and matching. The rim consistency term has only been used in the surface deformation process for the *roman* and *skull* datasets, for which accurate contours are available. The bounding volume information has *not* been used to filter out erroneous matches in our experiments. Our algorithm has successfully reconstructed various surface structures such as the high-curvature and/or shallow surface details of *roman*, the thin cheek bone and deep eye sockets of *skull*, and the intricate facial features of *face* and *face-2*. Quantitative comparisons kindly provided by D. Scharstein on the datasets presented in [20] show that the proposed method outperforms all the other evaluated techniques in terms of *accuracy* (distance d such that a given percentage of the reconstruction is within d from the ground truth model) and *completeness* (percentage of the ground truth model that is within a given distance from the reconstruction) on four out of the six datasets. The datasets consists of two objects (*temple* and *dino*), each of which constitutes three datasets (*sparse ring, ring,* and *full*) with different numbers of input images, ranging from 16 to more than 300, and our method achieves the best accuracy and completeness on all the *dino* datasets and the smallest *sparse ring temple*. Note that the *sparse ring temple* and *dino* datasets consisting of 16 views have been shown in Fig. 7 and their quantitative comparison with the top performers [4, 5, 7, 18, 21, 22, 23] are given in Fig. 8. [6] Finally, the bottom part of Fig. 8 compares our algorithm with Hernández Esteban's method [7], which is one of the best multi-view stereo reconstruction algorithms today, for the *polynesian* dataset, where a laser scanned model is used as a ground truth. As shown by the close-ups in this figure, our model is qualitatively better than the Herández's model, especially at sharp concave structures. This is also shown quantitatively using the same accuracy and completeness measures as before.

Reconstruction results for *scene* datasets are shown in Fig. 9. Additional information (such as segmentation masks, bounding boxes, or valid depth ranges) is not available in this case. The *city-hall* example is interesting because viewpoints change significantly across input cameras, and part of the building is only visible in some of the frames. Nonetheless, our algorithm has successfully reconstructed the whole scene with fine structural details. The *wall* dataset is challenging since a large portion of several of the input pictures consists of running water, and the corresponding image regions have successfully been detected as outliers, while accurate surface details have been recovered for the rigid wall structure. Finally, Fig. 10 illustrates our results on *crowded scene* datasets. Our algorithm reconstructs the background building from the *brussels* dataset, despite people occluding various parts of the scene. The *steps-2* dataset is an artificially generated example, where we have manually painted a red cartoonish human in each image of *steps* images. To further test the robustness of our algorithm against outliers, the *steps-3* dataset has been created from *steps-2* by copying its images but replacing the fifth one with the third, without changing camera parameters. This is a particularly challenging example, since the whole fifth image must be detected as an outlier. We have successfully reconstructed the details of both despite these outliers. Note

[6] Rendered views of the reconstructions and all the quantitative evaluations can be found at http://vision.middlebury.edu/mview/.

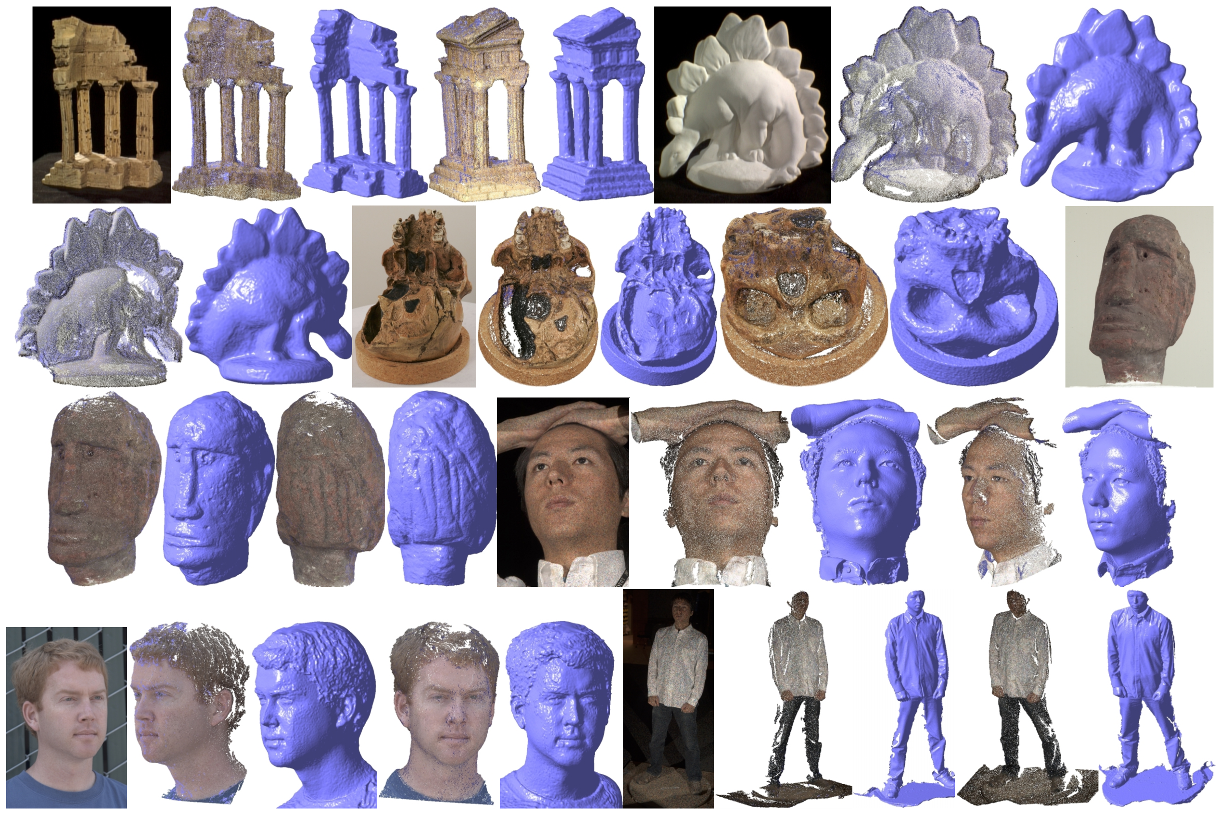


Figure 7. Sample results on object datasets: From left to right and top to bottom: *temple*, *dino*, *skull*, *polynesian*, *face*, *face-2*, and *body* datasets. In each case, one of the input image is shown, along with two views of texture-mapped reconstructed patches and shaded polygonal surfaces.

that the convex hull of the reconstructed patches' centers is used for the surface initialization except for the *city-hall* and *brussels*, for which the union of hemispheres is used.

The bottleneck of our multi-view stereo matching algorithm is the patch expansion step, whose running time varies from about 20 minutes, for small datasets such as *temple* and *dino*, to up to a few hours for datasets consisting of high-resolution images, such as *polynesian* and *city-hall*. The running times of polygonal surface extraction also range from 30 minutes to a few hours depending on the size of datasets. This is comparable to many variational methods [20], despite the fact that our algorithm does not involve any large optimization problem. This is due to several factors: First, unlike algorithms using voxels or discretized depth labels, our method solves a fully continuous optimization problem, thus does not suffer from discretization errors and can handle high-resolution input images directly, but trades speed for accuracy. Second, we use a region-based photometric consistency measure, which is much slower than a point-based measure, but takes into account surface orientation during optimization. In turn, this allows our algorithm to handle gracefully outdoor images with varying illumination. Again, accuracy and speed are conflicting requirements. To conclude, let us note that our future work will be aimed at adding a temporal component to our reconstruction algorithm, with the aim of achieving markerless face and body motion capture.

Acknowledgments: This work was supported in part by the National Science Foundation under grant IIS-0535152. We thank S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski for the *temple* and *dino* datasets and evaluations, C. Hernández Esteban, F. Schmitt, and the Museum of Cherbourg for *polynesian*, S. Sullivan, A. Suter, and Industrial Light and Magic for *face*, *face-2*, *body*, *steps*, and *wall*, C. Strecha for *city-hall* and *brussels*, and finally J. Blumenfeld and S. R. Leigh for the *skull* data set.

References

[1] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods and the stereo problem. *IEEE Trans. Im. Proc.*, 7(3):336–344, 1998. 1, 2

[2] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation by image exploration. In *ECCV*, 2004. 2

[3] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. In *ECCV*, volume 1, 2006. 1, 2

[4] Y. Furukawa and J. Ponce. High-fidelity image-based modeling. Technical Report CVR 2006-02, University of Illinois at Urbana-Champaign, 2006. 1, 2, 5, 6

[5] M. Goesele, B. Curless, and S. M. Seitz. Multi-view stereo revisited. In *CVPR*, pages 2402–2409, 2006. 2, 6

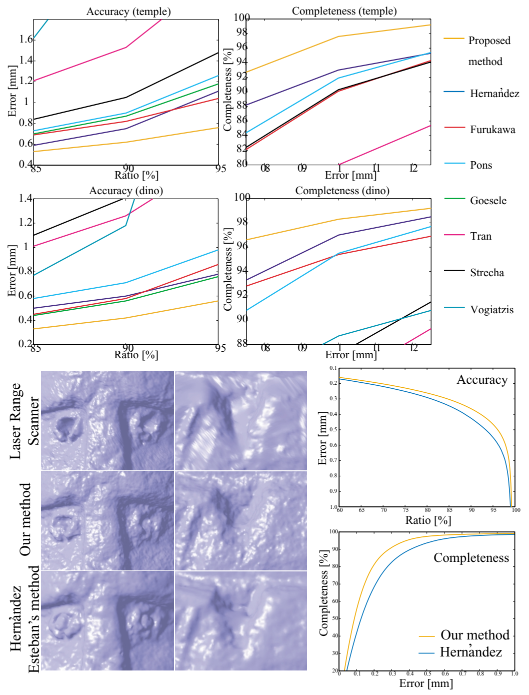


Figure 8. Quantitative comparison with other multi-view stereo algorithms for the *temple* and *dino* at the top, and for the *polynesian* at the bottom.

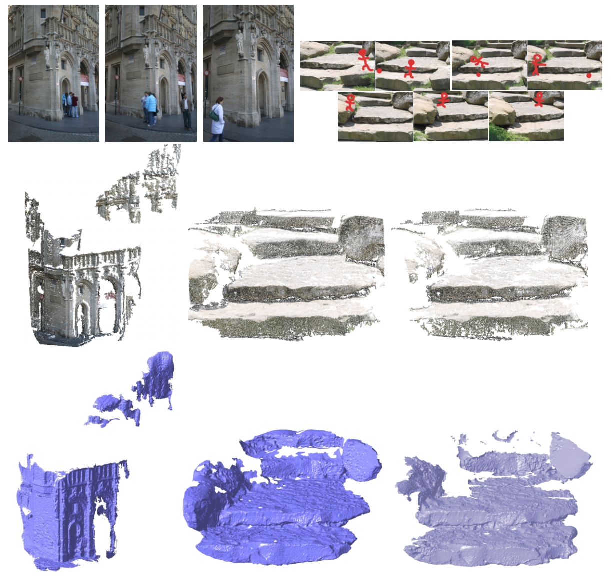


Figure 10. Sample results on *crowded scene* datasets. Top row: input images for *brussels* and *steps-2*; Second and third rows: reconstruction results for *brussels*, *steps-2*, and *steps-3*. Note that the *steps-3* dataset is generated by copying *steps-2* but replacing its third image by the fifth without changing camera parameters.

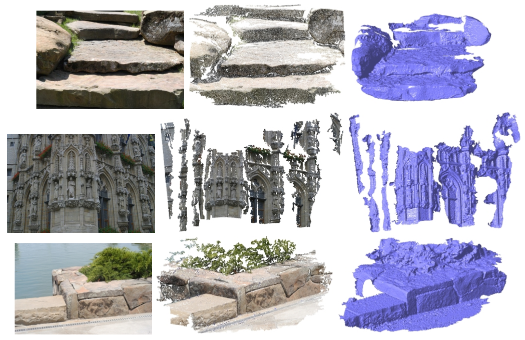


Figure 9. Sample results on *scene* datasets. From top to bottom: *steps*, *city-hall*, and *wall* datasets.

[6] M. Habbecke and L. Kobbelt. Iterative multi-view plane fitting. In *11th Fall Workshop on VISION, MODELING, AND VISUALIZATION*, 2006. 1, 2

[7] C. Hernández Esteban and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *CVIU*, 96(3), 2004. 1, 2, 6

[8] A. Hornung and L. Kobbelt. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *CVPR*, 2006. 1, 2

[9] R. Keriven. A variational framework to shape from contours. Technical Report 2002-221, ENPC, 2002. 1, 2

[10] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004. 5

[11] A. Kushal and J. Ponce. A novel approach to modeling 3d objects from stereo views and recognizing them in photographs. In *ECCV*, volume 2, pages 563–574, 2006. 2

[12] K. Kutulakos and S. Seitz. A theory of shape by space carving. *IJCV*, 38(3):199–218, 2000. 1, 2

[13] M. Lhuillier and L. Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *PAMI*, 27(3):418–433, 2005. 1, 2

[14] W. Naylor and B. Chapman. Wnlib. 3, 6

[15] M. Okutami and T. Kanade. A multiple-baseline stereo system. *PAMI*, 15(4):353–363, 1993. 1, 2

[16] G. P. Otto and T. K. W. Chau. 'region-growing' algorithm for matching of terrain images. *Image Vision Comput.*, 7(2):83–94, 1989. 2

[17] S. Paris, F. Sillion, and L. Quan. A surface reconstruction method using global graph cut optimization. In *ACCV*, January 2004. 1, 2

[18] J.-P. Pons, R. Keriven, and O. D. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *CVPR (2)*, pages 822–827, 2005. 1, 2, 6

[19] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. In *CVPR*, pages 1067–1073, 1997. 1, 2

[20] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *CVPR*, 1, 2006. 1, 6, 7

[21] C. Strecha, R. Fransens, and L. V. Gool. Combined depth and outlier estimation in multi-view stereo. In *CVPR*, pages 2394–2401, 2006. 1, 2, 6

[22] S. Tran and L. Davis. 3d surface reconstruction using graph cuts with surface constraints. In *ECCV*, 2006. 1, 2, 6

[23] G. Vogiatzis, P. H. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *CVPR*, 2005. 1, 2, 6

MVSNet: Depth Inference for Unstructured Multi-view Stereo

Yao Yao[1], Zixin Luo[1], Shiwei Li[1], Tian Fang[2], and Long Quan[1]

[1] The Hong Kong University of Science and Technology,
{yyaoag, zluoag, slibc, quan}@cse.ust.hk
[2] Shenzhen Zhuke Innovation Technology (Altizure),
fangtian@altizure.com

Abstract. We present an end-to-end deep learning architecture for depth map inference from multi-view images. In the network, we first extract deep visual image features, and then build the 3D cost volume upon the reference camera frustum via the differentiable homography warping. Next, we apply 3D convolutions to regularize and regress the initial depth map, which is then refined with the reference image to generate the final output. Our framework flexibly adapts arbitrary N-view inputs using a variance-based cost metric that maps multiple features into one cost feature. The proposed MVSNet is demonstrated on the large-scale indoor *DTU* dataset. With simple post-processing, our method not only significantly outperforms previous state-of-the-arts, but also is several times faster in runtime. We also evaluate MVSNet on the complex outdoor *Tanks and Temples* dataset, where our method ranks first before April 18, 2018 without any fine-tuning, showing the strong generalization ability of MVSNet.

Keywords: Multi-view Stereo, Depth Map, Deep Learning

1 Introduction

Multi-view stereo (MVS) estimates the dense representation from overlapping images, which is a core problem of computer vision extensively studied for decades. Traditional methods use hand-crafted similarity metrics and engineered regularizations (e.g., normalized cross correlation and semi-global matching [12]) to compute dense correspondences and recover 3D points. While these methods have shown great results under ideal Lambertian scenarios, they suffer from some common limitations. For example, low-textured, specular and reflective regions of the scene make dense matching intractable and thus lead to incomplete reconstructions. It is reported in recent MVS benchmarks [1,18] that, although current state-of-the-art algorithms [7,36,8,32] perform very well on the *accuracy*, the reconstruction *completeness* still has large room for improvement.

Recent success on convolutional neural networks (CNNs) research has also triggered the interest to improve the stereo reconstruction. Conceptually, the learning-based method can introduce global semantic information such as specular and reflective priors for more robust matching. There are some attempts on the two-view stereo matching, by replacing either hand-crafted similarity metrics [39,10,23,11] or engineered regularizations [34,19,17] with the learned ones. They have shown promising results and gradually surpassed traditional methods in stereo benchmarks [9,25]. In fact, the stereo matching task is perfectly suitable for applying CNN-based methods, as image pairs are rectified in advance and thus the problem becomes the horizontal pixel-wise disparity estimation without bothering with camera parameters.

However, directly extending the learned two-view stereo to multi-view scenarios is non-trivial. Although one can simply pre-rectify all selected image pairs for stereo matching, and then merge all pairwise reconstructions to a global point cloud, this approach fails to fully utilize the multi-view information and leads to less accurate result. Unlike stereo matching, input images to MVS could be of arbitrary camera geometries, which poses a tricky issue to the usage of learning methods. Only few works acknowledge this problem and try to apply CNN to the MVS reconstruction: SurfaceNet [14] constructs the Colored Voxel Cubes (CVC) in advance, which combines all image pixel color and camera information to a single volume as the input of the network. In contrast, the Learned Stereo Machine (LSM) [15] directly leverages the differentiable projection/unprojection to enable the end-to-end training/inference. However, both the two methods exploit the volumetric representation of regular grids. As restricted by the huge memory consumption of 3D volumes, their networks can hardly be scaled up: LSM only handles synthetic objects in low volume resolution, and SurfaceNet applies a heuristic divide-and-conquer strategy and takes a long time for large-scale reconstructions. For the moment, the leading boards of modern MVS benchmarks are still occupied by traditional methods [7,8,32].

To this end, we propose an end-to-end deep learning architecture for depth map inference, which computes one depth map at each time, rather than the whole 3D scene at once. Similar to other depth map based MVS methods [35,3,8,32], the proposed network, MVSNet, takes one reference image and several source images as input, and infers the depth map for the reference image. The key insight here is the differentiable homography warping operation, which implicitly encodes camera geometries in the network to build the 3D cost volumes from 2D image features and enables the end-to-end training. To adapt arbitrary number of source images in the input, we propose a variance-based metric that maps multiple features into one cost feature in the volume. This cost volume then undergoes multi-scale 3D convolutions and regress an initial depth map. Finally, the depth map is refined with the reference image to improve the accuracy of boundary areas. There are two major differences between our method and previous learned approaches [15,14]. First, for the purpose of depth map inference, our 3D cost volume is built upon the camera frustum instead of the regular Euclidean space. Second, our method decouples the MVS reconstruction to smaller problems of per-view depth map estimation, which makes large-scale reconstruction possible.

We train and evaluate the proposed MVSNet on the large-scale *DTU* dataset [1]. Extensive experiments show that with simple post-processing, MVSNet outperforms all competing methods in terms of completeness and overall quality. Besides, we demonstrate the generalization power of the network on the outdoor *Tanks and Temples* benchmark [18], where MVSNet ranks first (before April. 18, 2018) over all submissions including the open-source MVS methods (e.g., COLMAP [32] and OpenMVS [29]) and commercial software (Pix4D [30]) without any fine-tuning. It is also noteworthy that the runtime of MVSNet is several times or even several orders of magnitude faster than previous state-of-the-arts.

2 Related work

MVS Reconstruction. According to output representations, MVS methods can be categorized into 1) direct point cloud reconstructions [22,7], 2) volumetric reconstructions [20,33,14,15] and 3) depth map reconstructions [35,3,8,32,38]. Point cloud based methods operate directly on 3D points, usually relying on the propagation strategy to gradually densify the reconstruction [22,7]. As the propagation of point clouds is proceeded sequentially, these methods are difficult to be fully parallelized and usually take a long time in processing. Volumetric based methods divide the 3D space into regular grids and then estimate if each voxel is adhere to the surface. The downsides for this representation are the space discretization error and the high memory consumption. In contrast, depth map is the most flexible representation among all. It decouples the complex MVS problem into relatively small problems of per-view depth map estimation, which focuses on only one reference and a few source images at a time. Also, depth maps can be easily fused to the point cloud [26] or the volumetric reconstructions [28]. According to the recent MVS benchmarks [1,18], current best MVS algorithms [8,32] are both depth map based approaches.

Learned Stereo. Rather than using traditional handcrafted image features and matching metrics [13], recent studies on stereo apply the deep learning technique for better pair-wise patch matching. Han *et al.* [10] first propose a deep network to match two image patches. Zbontar *et al.* [39] and Luo *et al.* [23] use the learned features for stereo matching and semi-global matching (SGM) [12] for post-processing. Beyond the pair-wise matching cost, the learning technique is also applied in cost regularization. SGMNet [34] learns to adjust the parameters used in SGM, while CNN-CRF [19] integrates the conditional random field optimization in the network for the end-to-end stereo learning. The recent state-of-the-art method is GCNet [17], which applies 3D CNN to regularize the cost volume and regress the disparity by the soft argmin operation. It has been reported in KITTI banchmark [25] that, learning-based stereos, especially those end-to-end learning algorithms [24,19,17], significantly outperform the traditional stereo approaches.

Learned MVS. There are fewer attempts on learned MVS approaches. Hartmann *et al.* propose the learned multi-patch similarity [11] to replace the traditional cost metric for MVS reconstruction. The first learning based pipeline for MVS problem is SurfaceNet [14], which pre-computes the cost volume with sophisticated voxel-wise view selection, and uses 3D CNN to regularize and infer

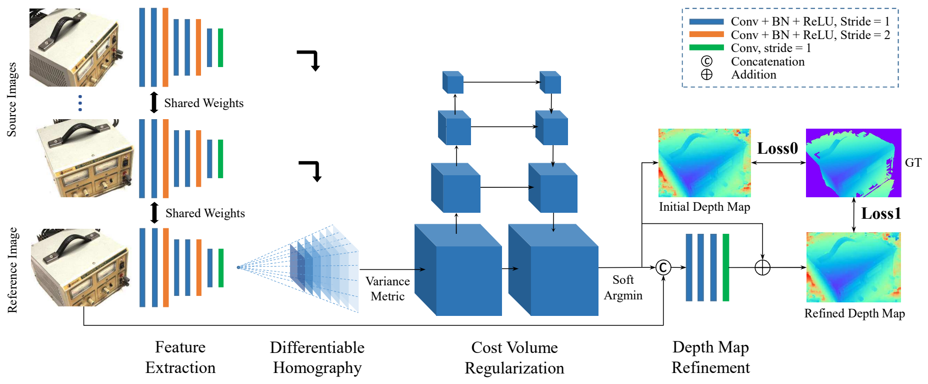


Fig. 1: The network design of MVSNet. Input images will go through the 2D feature extraction network and the differentiable homograph warping to generate the cost volume. The final depth map output is regressed from the regularized probability volume and refined with the reference image

the surface voxels. The most related approach to ours is the LSM [15], where camera parameters are encoded in the network as the projection operation to form the cost volume, and 3D CNN is used to classify if a voxel belongs to the surface. However, due to the common drawback of the volumetric representation, networks of SurfaceNet and LSM are restricted to only small-scale reconstructions. They either apply the divide-and-conquer strategy [14] or is only applicable to synthetic data with low resolution inputs [15]. In contrast, our network focus on producing the depth map for one reference image at each time, which allows us to adaptively reconstruct a large scene directly.

3 MVSNet

This section describes the detailed architecture of the proposed network. The design of MVSNet strongly follows the rules of camera geometry and borrows the insights from previous MVS approaches. In following sections, we will compare each step of our network to the traditional MVS methods, and demonstrate the advantages of our learning-based MVS system. The full architecture of MVSNet is visualized in Fig. 1.

3.1 Image Features

The first step of MVSNet is to extract the deep features $\{\mathbf{F}_i\}_{i=1}^N$ of the N input images $\{\mathbf{I}_i\}_{i=1}^N$ for dense matching. An eight-layer 2D CNN is applied, where the strides of layer 3 and 6 are set to two to divide the feature towers into three scales. Within each scale, two convolutional layers are applied to extract the higher-level image representation. Each convolutional layer is followed by a batch-normalization (BN) layer and a rectified linear unit (ReLU) except for the last layer. Also, similar to common matching tasks, parameters are shared among all feature towers for efficient learning.

The outputs of the 2D network are N 32-channel feature maps downsized by four in each dimension compared with input images. It is noteworthy that though the image frame is downsized after feature extraction, the original neighboring information of each remaining pixel has already been encoded into the 32-channel pixel descriptor, which prevents dense matching from losing useful context information. Compared with simply performing dense matching on original images, the extracted feature maps significantly boost the reconstruction quality (see Sec. 5.3).

3.2 Cost Volume

The next step is to build a 3D cost volume from the extracted feature maps and input cameras. While previous works [14,15] divide the space using regular grids, for our task of depth map inference, we construct the cost volume upon the reference camera frustum. For simplicity, in the following we denote $\mathbf{I}_1$ as the reference image, $\{\mathbf{I}_i\}_{i=2}^N$ the source images, and $\{\mathbf{K}_i, \mathbf{R}_i, \mathbf{t}_i\}_{i=1}^N$ the camera intrinsics, rotations and translations that correspond to the feature maps.

Differentiable Homography All feature maps are warped into different fronto-parallel planes of the reference camera to form N feature volumes $\{\mathbf{V}_i\}_{i=1}^N$. The coordinate mapping from the warped feature map $\mathbf{V}_i(d)$ to $\mathbf{F}_i$ at depth d is determined by the planar transformation $\mathbf{x}' \sim \mathbf{H}_i(d) \cdot \mathbf{x}$, where '$\sim$' denotes the projective equality and $\mathbf{H}_i(d)$ the homography between the i^{th} feature map and the reference feature map at depth d. Let $\mathbf{n}_1$ be the principle axis of the reference camera, the homography is expressed by a 3×3 matrix:

$$\mathbf{H}_i(d) = \mathbf{K}_i \cdot \mathbf{R}_i \cdot \left(\mathbf{I} - \frac{(\mathbf{t}_1 - \mathbf{t}_i) \cdot \mathbf{n}_1^T}{d}\right) \cdot \mathbf{R}_1^T \cdot \mathbf{K}_1^T. \tag{1}$$

Without loss of generality, the homography for reference feature map $\mathbf{F}_1$ itself is an 3×3 identity matrix. The warping process is similar to that of the classical plane sweeping stereo [5], except that the differentiable bilinear interpolation is used to sample pixels from feature maps $\{\mathbf{F}_i\}_{i=1}^N$ rather than images $\{\mathbf{I}_i\}_{i=1}^N$. As the core step to bridge the 2D feature extraction and the 3D regularization networks, the warping operation is implemented in differentiable manner, which enables end-to-end training of depth map inference.

Cost Metric Next, we aggregate multiple feature volumes $\{\mathbf{V}_i\}_{i=1}^N$ to one cost volume $\mathbf{C}$. To adapt arbitrary number of input views, we propose a variance-based cost metric $\mathcal{M}$ for N-view similarity measurement. Let W, H, D, F be the input image width, height, depth sample number and the channel number of the feature map, and $V = \frac{W}{4} \cdot \frac{H}{4} \cdot D \cdot F$ the feature volume size, our cost metric defines the mapping $\mathcal{M} : \underbrace{\mathbb{R}^V \times \cdots \times \mathbb{R}^V}_{N} \rightarrow \mathbb{R}^V$ that:

$$\mathbf{C} = \mathcal{M}(\mathbf{V}_1, \cdots, \mathbf{V}_N) = \frac{\sum\limits_{i=1}^{N} (\mathbf{V}_i - \overline{\mathbf{V}_i})^2}{N} \tag{2}$$

Where $\overline{\mathbf{V}_i}$ is the average volume among all feature volumes, and all operations above are element-wise.

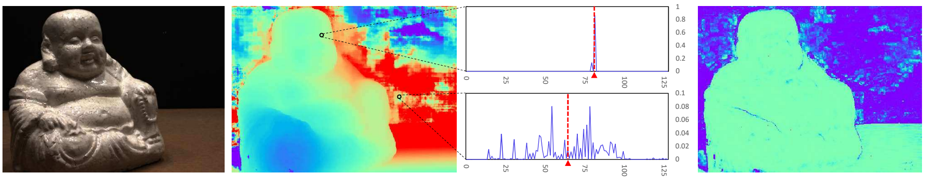
Most traditional MVS methods aggregate pairwise costs between the reference image and all source images in a heuristic way. Instead, our metric design follows the philosophy that all views should contribute equally to the matching cost and gives no preference to the reference image [11]. We notice that recent work [11] applies the mean operation with multiple CNN layers to infer the multi-patch similarity. Here we choose the 'variance' operation instead because the 'mean' operation itself provides no information about the feature differences, and their network requires pre- and post- CNN layers to help infer the similarity. In contrast, our variance-based cost metric explicitly measures the multi-view feature difference. In later experiments, we will show that such explicit difference measurement improves the validation accuracy.

Cost Volume Regularization The raw cost volume computed from image features could be noise-contaminated (e.g., due to the existence of non-Lambertian surfaces or object occlusions) and should be incorporated with smoothness constraints to infer the depth map. Our regularization step is designed for refining the above cost volume $\mathbf{C}$ to generate a probability volume $\mathbf{P}$ for depth inference. Inspired by recent learning-based stereo [17] and MVS [14,15] methods, we apply the multi-scale 3D CNN for cost volume regularization. The four-scale network here is similar to a 3D version UNet [31], which uses the encoder-decoder structure to aggregate neighboring information from a large receptive field with relatively low memory and computation cost. To further lessen the computational requirement, we reduce the 32-channel cost volume to 8-channel after the first 3D convolutional layer, and change the convolutions within each scale from 3 layers to 2 layers. The last convolutional layer outputs a 1-channel volume. We finally apply the *softmax* operation along the depth direction for probability normalization.

The resulting probability volume is highly desirable in depth map inference that it can not only be used for per-pixel depth estimation, but also for measuring the estimation confidence. We will show in Sec. 3.3 that one can easily determine the depth reconstruction quality by analyzing its probability distribution, which leads to a very concise yet effective outlier filtering strategy in Sec. 4.2.

3.3 Depth Map

Initial Estimation The simplest way to retrieve depth map $\mathbf{D}$ from the probability volume $\mathbf{P}$ is the pixel-wise winner-take-all [5] (i.e., *argmax*). However,



(a) Reference image (b) Inferred depth map (c) Probability distribution (d) Probability Map

Fig. 2: Illustrations on inferred depth map, probability distributions and probability map. (a) One reference image of *scan* 114, *DTU* dataset [1]; (b) the inferred depth map; (c) the probability distributions of an inlier pixel (top) and an outlier pixel (bottom), where the x-axis is the index of depth hypothesis, y-axis the probability and red lines the soft argmin results; (d) the probability map. As shown in (c), the outlier's distribution is scattered and results in a low probability estimation in (d)

the *argmax* operation is unable to produce sub-pixel estimation, and cannot be trained with back-propagation due to its indifferentiability. Instead, we compute the *expectation* value along the depth direction, i.e., the probability weighted sum over all hypotheses:

$$\mathbf{D} = \sum_{d=d_{min}}^{d_{max}} d \times \mathbf{P}(d) \tag{3}$$

Where $\mathbf{P}(d)$ is the probability estimation for all pixels at depth d. Note that this operation is also referred to as the *soft argmin* operation in [17]. It is fully differentiable and able to approximate the argmax result. While the depth hypotheses are uniformly sampled within range $[d_{min}, d_{max}]$ during cost volume construction, the expectation value here is able to produce a continuous depth estimation. The output depth map (Fig. 2 (b)) is of the same size to 2D image feature maps, which is downsized by four in each dimension compared to input images.

Probability Map The probability distribution along the depth direction also reflects the depth estimation quality. Although the multi-scale 3D CNN has very strong ability to regularize the probability to the single modal distribution, we notice that for those falsely matched pixels, their probability distributions are scattered and cannot be concentrated to one peak (see Fig. 2 (c)). Based on this observation, we define the quality of a depth estimation $\hat{d}$ as the probability that the ground truth depth is within a small range near the estimation. As depth hypotheses are discretely sampled along the camera frustum, we simply take the probability sum over the four nearest depth hypotheses to measure the estimation quality. Notice that other statistical measurements, such as standard deviation or entropy can also be used here, but in our experiments we observe no significant improvement from these measurements for depth map filtering. Moreover, our probability sum formulation leads to a better control of thresholding parameter for outliers filtering.

Depth Map Refinement While the depth map retrieved from the probability volume is a qualified output, the reconstruction boundaries may suffer from oversmoothing due to the large receptive field involved in the regularization, which is similar to the problems in semantic segmentation [4] and image matting [37]. Notice that the reference image in natural contains boundary information, we thus use the reference image as a guidance to refine the depth map. Inspired by the recent image matting algorithm [37], we apply a depth residual learning network at the end of MVSNet. The initial depth map and the resized reference image are concatenated as a 4-channel input, which is then passed through three 32-channel 2D convolutional layers followed by one 1-channel convolutional layer to learn the depth residual. The initial depth map is then added back to generate the refined depth map. The last layer does not contain the BN layer and the ReLU unit as to learn the negative residual. Also, to prevent being biased at a certain depth scale, we pre-scale the initial depth magnitude to range [0, 1], and convert it back after the refinement.

3.4 Loss

Losses for both the initial depth map and the refined depth map are considered. We use the mean absolute difference between the ground truth depth map and the estimated depth map as our training loss. As ground truth depth maps are not always complete in the whole image (see Sec. 4.1), we only consider those pixels with valid ground truth labels:

$$Loss = \sum_{p \in \mathbf{p}_{valid}} \underbrace{\| d(p) - \hat{d_i}(p) \|_1}_{Loss0} + \lambda \cdot \underbrace{\| d(p) - \hat{d_r}(p) \|_1}_{Loss1} \tag{4}$$

Where $\mathbf{p}_{valid}$ denotes the set of valid ground truth pixels, $d(p)$ the ground truth depth value of pixel p, $\hat{d_i}(p)$ the initial depth estimation and $\hat{d_r}(p)$ the refined depth estimation. The parameter λ is set to 1.0 in experiments.

4 Implementations

4.1 Training

Data Preparation Current MVS datasets provide ground truth data in either point cloud or mesh formats, so we need to generate the ground truth depth maps ourselves. The *DTU* dataset [1] is a large-scale MVS dataset containing more than 100 scenes with different lighting conditions. As it provides the ground truth point cloud with normal information, we use the screened Poisson surface reconstruction (SPSR) [16] to generate the mesh surface, and then render the mesh to each viewpoint to generate the depth maps for our training. The parameter, *depth-of-tree* is set to 11 in SPSR to acquire the high quality mesh result. Also, we set the mesh *trimming-factor* to 9.5 to alleviate mesh artifacts in surface edge areas. To fairly compare MVSNet with other learning based methods, we choose the same training, validation and evaluation sets as in SurfaceNet [14][1]. Considering each scan contains 49 images with 7 different lighting conditions, by setting each image as the reference, *DTU* dataset provides 27097 training samples in total.

View Selection A reference image and two source images ($N = 3$) are used in our training. We calculate a score $s(i, j) = \sum_{\mathbf{p}} \mathcal{G}(\theta_{ij}(\mathbf{p}))$ for each image pair according to the sparse points, where $\mathbf{p}$ is a common track in both view i and j, $\theta_{ij}(\mathbf{p}) = (180/\pi)\arccos((\mathbf{c}_i - \mathbf{p}) \cdot (\mathbf{c}_j - \mathbf{p}))$ is $\mathbf{p}$'s baseline angle and $\mathbf{c}$ is the camera center. $\mathcal{G}$ is a piecewise Gaussian function [40] that favors a certain baseline angle θ_0:

$$\mathcal{G}(\theta) = \begin{cases} \exp(-\frac{(\theta-\theta_0)^2}{2\sigma_1^2}), \theta \leq \theta_0 \\ \exp(-\frac{(\theta-\theta_0)^2}{2\sigma_2^2}), \theta > \theta_0 \end{cases}$$

In the experiments, θ_0, σ_1 and σ_2 are set to 5, 1 and 10 respectively.

Notice that images will be downsized in feature extraction, plus the four-scale encoder-decoder structure in 3D regularization part, the input image size must be divisible by a factor of 32. Considering this requirement also the limited GPU memories, we downsize the image resolution from 1600×1200 to 800×600, and then crop the image patch with $W = 640$ and $H = 512$ from the center as the training input. The input camera parameters are changed accordingly. The depth hypotheses are uniformly sampled from $425mm$ to $935mm$ with a $2mm$ resolution ($D = 256$). We use TensorFlow [2] to implement MVSNet, and the network is trained on one Tesla P100 graphics card for around $100,000$ iterations.

4.2 Post-processing

Depth Map Filter The above network estimates a depth value for every pixel. Before converting the result to dense point clouds, it is necessary to filter out outliers at those background and occluded areas. We propose two criteria, namely *photometric* and *geometric* consistencies for the robust depth map filtering.

The photometric consistency measures the matching quality. As discussed in Sec. 3.3, we compute the probability map to measure the depth estimation quality. In our experiments, we regard pixels with probability lower than 0.8 as outliers. The geometric constraint measures the depth consistency among multiple views. Similar to the left-right disparity check for stereo, we project a reference pixel p_1 through its depth d_1 to pixel p_i in another view, and then reproject p_i back to the reference image by p_i's depth estimation d_i. If the reprojected coordinate p_{reproj} and and the reprojected depth d_{reproj} satisfy $|p_{reproj} - p_1| < 1$ and $|d_{reproj} - d_1|/d_1 < 0.01$, we say the depth estimation d_1 of p_1 is two-view consistent. In our experiments, all depths should be at least three view consistent. This simple two-step filtering strategy shows strong robustness for filtering different kinds of outliers.

[1] Validation set: scans {3, 5, 17, 21, 28, 35, 37, 38, 40, 43, 56, 59, 66, 67, 82, 86, 106, 117}. Evaluation set: scans {1, 4, 9, 10, 11, 12, 13, 15, 23, 24, 29, 32, 33, 34, 48, 49, 62, 75, 77, 110, 114, 118}. Training set: the other 79 scans.

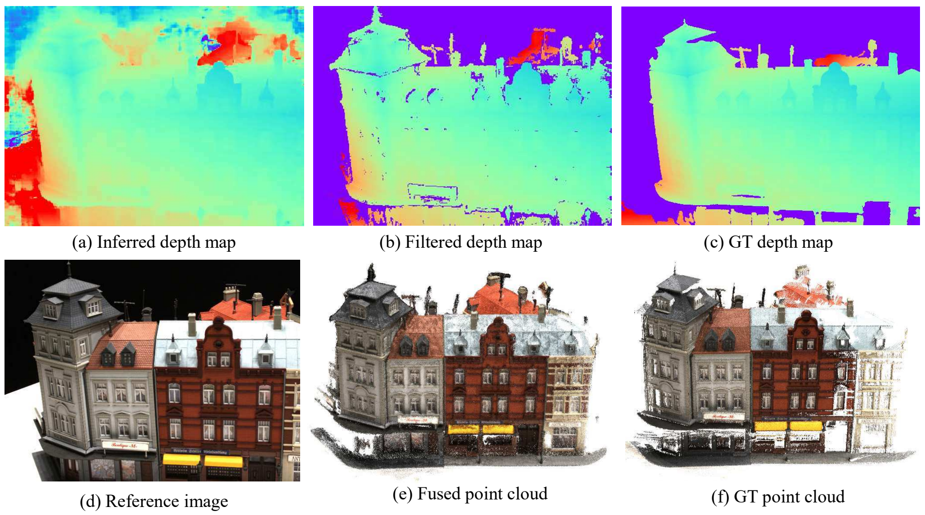


Fig. 3: Reconstructions of *scan* 9, *DTU* dataset [1]. From top left to bottom right: (a) the inferred depth map from MVSNet; (b) the filtered depth map after photometric and geometric filtering; (c) the depth map rendered from the ground truth mesh; (d) the reference image; (e) the final fused point cloud; (f) the ground truth point cloud

Depth Map Fusion Similar to other multi-view stereo methods [8,32], we apply a depth map fusion step to integrate depth maps from different views to a unified point cloud representation. The visibility-based fusion algorithm [26] is used in our reconstruction, where depth occlusions and violations across different viewpoints are minimized. To further suppress reconstruction noises, we determine the visible views for each pixel as in the filtering step, and take the average over all reprojected depths $\overline{d_{reproj}}$ as the pixel's final depth estimation. The fused depth maps are then directly reprojected to space to generate the 3D point cloud. The illustration of our MVS reconstruction is shown in Fig. 3.

5 Experiments

5.1 Benchmarking on *DTU* dataset

We first evaluate our method on the 22 evaluation scans of the *DTU* dataset [1]. The input view number, image width, height and depth sample number are set to $N = 5$, $W = 1600$, $H = 1184$ and $D = 256$ respectively. For quantitative evaluation, we calculate the *accuracy* and the *completeness* of both the distance metric [1] and the percentage metric [18]. While the matlab code for the distance metric is given by *DTU* dataset, we implement the percentage evaluation ourselves. Notice that the percentage metric also measures the overall performance of accuracy and completeness as the *f-score*. To give a similar measurement for

Table 1: Quantitative results on the *DTU*'s evaluation set [1]. We evaluate all methods using both the distance metric [1] (lower is better), and the percentage metric [18] (higher is better) with respectively $1mm$ and $2mm$ thresholds

	Mean Distance (mm)			Percentage ($<1mm$)			Percentage ($<2mm$)		
	Acc.	Comp.	*overall*	Acc.	Comp.	*f-score*	Acc.	Comp.	*f-score*
Camp [3]	0.835	0.554	0.695	71.75	64.94	66.31	84.83	67.82	73.02
Furu [7]	0.613	0.941	0.777	69.55	61.52	63.26	78.99	67.88	70.93
Tola [35]	0.342	1.190	0.766	90.49	57.83	68.07	93.94	63.88	73.61
Gipuma [8]	**0.283**	0.873	0.578	**94.65**	59.93	70.64	**96.42**	63.81	74.16
SurfaceNet[14]	0.450	1.04	0.745	83.8	63.38	69.95	87.15	67.99	74.4
MVSNet (Ours)	0.396	**0.527**	**0.462**	86.46	**71.13**	**75.69**	91.06	**75.31**	**80.25**

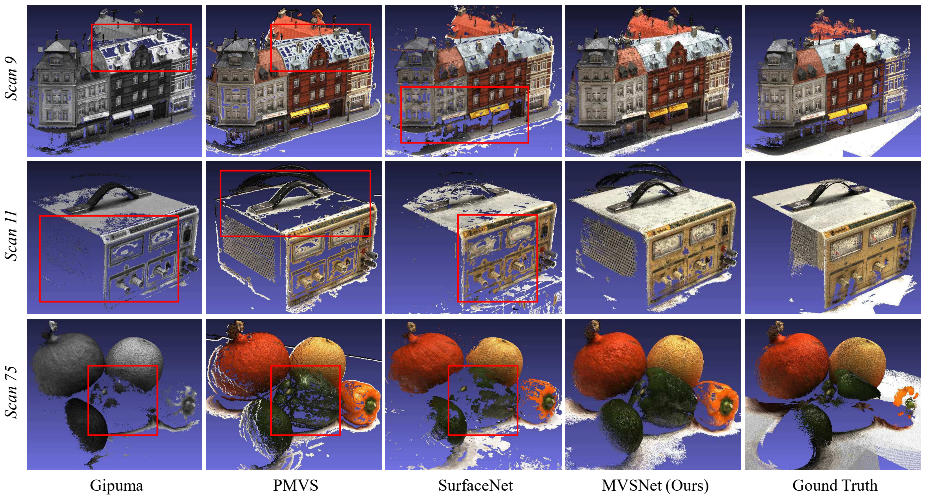


Fig. 4: Qualitative results of *scans* 9, 11 and 75 of *DTU* dataset [1]. Our MVSNet generates the most complete point clouds especially in those textureless and reflective areas. **Best viewed on screen**

the distance metric, we define the *overall score*, and take the average of mean accuracy and mean completeness as the reconstruction quality.

Quantitative results are shown in Table 1. While Gipuma [35] performs best in the accuracy, our MVSNet outperforms all methods in both the completeness and the overall quality **with a significant margin**. As shown in Fig. 4, MVSNet produces the most complete point clouds especially in those textureless and reflected areas, which are commonly considered as the most difficult parts to recover in MVS reconstruction.

Table 2: Quantitative results on *Tanks and Temples* benchmark [18]. MVSNet achieves best *f-score* result among all submissions without any fine-tuning

Method	Rank	Mean	Family	Francis	Horse	Lighthouse	M60	Panther	Playground	Train
MVSNet (Ours)	**3.00**	**43.48**	55.99	28.55	25.07	50.79	**53.96**	**50.86**	47.90	34.69
Pix4D [30]	3.12	43.24	**64.45**	31.91	**26.43**	54.41	50.58	35.37	47.78	34.96
COLMAP [32]	3.50	42.14	50.41	22.25	25.63	**56.43**	44.83	46.97	**48.53**	**42.04**
OpenMVG [27] + OpenMVS [29]	3.62	41.71	58.86	**32.59**	26.25	43.12	44.73	46.85	45.97	35.27
OpenMVG [27] + MVE [6]	6.00	38.00	49.91	28.19	20.75	43.35	44.51	44.76	36.58	35.95
OpenMVG [27] + SMVS [21]	10.38	30.67	31.93	19.92	15.02	39.38	36.51	41.61	35.89	25.12
OpenMVG-G [27] + OpenMVS [29]	10.88	22.86	56.50	29.63	21.69	6.55	39.54	28.48	0.00	0.53
MVE [6]	11.25	25.37	48.59	23.84	12.70	5.07	39.62	38.16	5.81	29.19
OpenMVG [27] + PMVS [7]	11.88	29.66	41.03	17.70	12.83	36.68	35.93	33.20	31.78	28.10

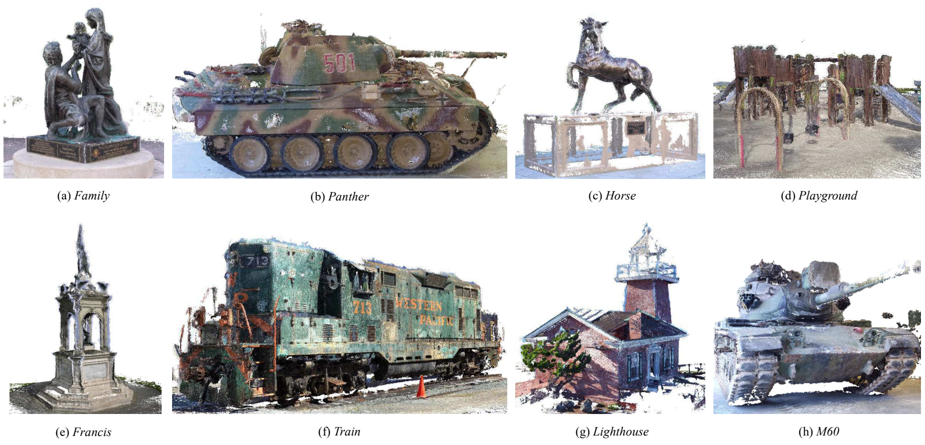
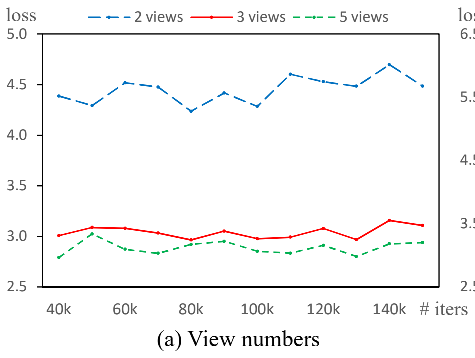


Fig. 5: Point cloud results of the *intermediate set* of *Tanks and Temples* [18] dataset, which demonstrates the generalization power of MVSNet on complex outdoor scenes

5.2 Generalization on *Tanks and Temples* dataset

The *DTU* scans are taken under well-controlled indoor environment with fixed camera trajectory. To further demonstrate the generalization ability of MVSNet, we test the proposed method on the more complex outdoor *Tanks and Temples* dataset [18], using the model trained on *DTU* **without any fine-tuning**. While we choose $N = 5$, $W = 1920$, $H = 1056$ and $D = 256$ for all reconstructions, the depth range and the source image set for the reference image are determined according to sparse point cloud and camera positions, which are recovered by the open source SfM software OpenMVG [27].

Our method ranks first before April 18, 2018 among all submissions of the *intermediate set* [18] according to the online benchmark (Table 2). Although the model is trained on the very different *DTU* indoor dataset, MVSNet is still able to produce the best reconstructions on these outdoor scenes, demonstrating



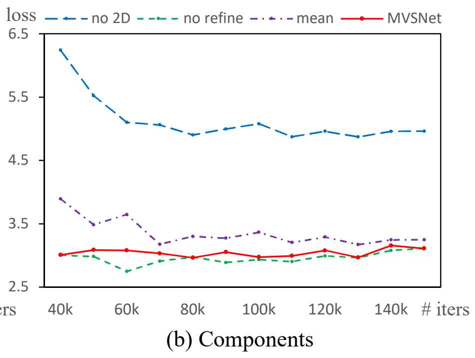


Fig. 6: Ablation studies. (a) Validation losses of different input view numbers. (b) Ablations on 2D image feature, cost metric and depth map refinement

the strong generalization ability of the proposed network. The qualitative point cloud results of the *intermediate set* are visualized in Fig. 5.

5.3 Ablations

This section analyzes several components in MVSNet. For all following studies, we use the validation loss to measure the reconstruction quality. The 18 validation scans (see Sec. 4.1) are pre-processed as the training set that we set $N = 3$, $W = 640$, $H = 512$ and $D = 256$ for the validation loss computation.

View Number We first study the influence of the input view number N and demonstrate that our model can be applied to arbitrary views of input. While the model in Sec. 4.1 is trained using $N = 3$ views, we test the model using $N = 2, 3, 5$ respectively. As expected, it is shown in Fig. 6 (a) that adding input views can lower the validation loss, which is consistent with our knowledge about MVS reconstructions. It is noteworthy that testing with $N = 5$ performs better than with $N = 3$, even though the model is trained with the 3 views setting. This highly desirable property makes MVSNet flexible enough to be applied the different input settings.

Image Features We demonstrate in this study that the learning based image feature could significantly boost the MVS reconstruction quality. To model the traditional patch-based image feature in MVSNet, we replace the original 2D feature extraction network with a single 32-channel convolutional layer. The filter kernel is set to a large number of 7×7 and the stride is set to 4. As shown in Fig. 6 (b), network with the 2D feature extraction significantly outperforms the single layer one on validation loss.

Cost Metric We also compare our variance operation based cost metric with the mean operation based metric [11]. The element-wise variance operation in

Eq. 2 is replaced with the mean operation to train the new model. It can be found in Fig. 6 (b) that our cost metric results in a faster convergence with lower validation loss, which demonstrates that it is more reasonable to use the explicit difference measurement to compute the multi-view feature similarity.

Depth Refinement Lastly, we train MVSNet with and without the depth map refinement network. The models are also tested on *DTU* evaluation set as in Sec. 5.1, and we use the percentage metric [18] to quantitatively compare the two models. While Fig. 6 (b) shows that the refinement does not affect the validation loss too much, the refinement network improves the evaluation results from 75.58 to 75.69 ($< 1mm$ *f-score*) and from 79.98 to 80.25 ($< 2mm$ *f-score*).

5.4 Discussions

Running Time We compare the running speed of MVSNet to Gipuma [8], COLMAP [32] and SurfaceNet [14] using the *DTU* evaluation set. The other methods are compiled from their source codes and all methods are tested in the same machine. MVSNet is much more efficient that it takes around 230 seconds to reconstruct one scan (**4.7 seconds** per view). The running speed is $\sim 5\times$ faster than Gipuma, $\sim 100\times$ than COLMAP and $\sim 160\times$ than SurfaceNet.

GPU Memory The GPU memory required by MVSNet is related to the input image size and the depth sample number. In order to test on the *Tanks and Temples* with the original image resolution and sufficient depth hypotheses, we choose the Tesla P100 graphics card (16 GB) to implement our method. It is noteworthy that the training and validation on *DTU* dataset could be done using one consumer level GTX 1080ti graphics card (11 GB).

Training Data As mentioned in Sec. 4.1, *DTU* provides ground truth point clouds with normal information so that we can convert them into mesh surfaces for depth maps rendering. However, currently *Tanks and Temples* dataset does not provide the normal information or mesh surfaces, so we are unable to fine-tune MVSNet on *Tanks and Temples* for better performance.

Although using such rendered depth maps have already achieved satisfactory results, some limitations still exist: 1) the provided ground truth meshes are not 100% complete, so some triangles behind the foreground will be falsely rendered to the depth map as the valid pixels, which may deteriorate the training process. 2) If a pixel is occluded in all other views, it should not be used for training. However, without the complete mesh surfaces we cannot correctly identify the occluded pixels. We hope future MVS datasets could provide ground truth depth maps with complete occlusion and background information.

6 Conclusion

We have presented a deep learning architecture for MVS reconstruction. The proposed MVSNet takes unstructured images as input, and infers the depth map for the reference image in an end-to-end fashion. The core contribution of MVSNet is to encode the camera parameters as the differentiable homography to build the cost volume upon the camera frustum, which bridges the 2D feature extraction and 3D cost regularization networks. It has been demonstrated on *DTU* dataset that MVSNet not only significantly outperforms previous methods, but also is more efficient in speed by several times. Also, MVSNet have produced the state-of-the-art results on *Tanks and Temples* dataset without any fine-tuning, which demonstrates its strong generalization ability.

References

1. Aanæs, H., Jensen, R.R., Vogiatzis, G., Tola, E., Dahl, A.B.: Large-scale data for multiple-view stereopsis. International Journal of Computer Vision (IJCV) (2016)
2. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`, software available from tensorflow.org
3. Campbell, N.D., Vogiatzis, G., Hernández, C., Cipolla, R.: Using multiple hypotheses to improve depth-maps for multi-view stereo. European Conference on Computer Vision (ECCV) (2008)
4. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2017)
5. Collins, R.T.: A space-sweep approach to true multi-image matching. Computer Vision and Pattern Recognition (CVPR) (1996)
6. Fuhrmann, S., Langguth, F., Goesele, M.: Mve-a multi-view reconstruction environment. Eurographics Workshop on Graphics and Cultural Heritage (GCH) (2014)
7. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multiview stereopsis. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2010)
8. Galliani, S., Lasinger, K., Schindler, K.: Massively parallel multiview stereopsis by surface normal diffusion. International Conference on Computer Vision (ICCV) (2015)
9. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. Computer Vision and Pattern Recognition (CVPR) (2012)
10. Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A.C.: Matchnet: Unifying feature and metric learning for patch-based matching. Computer Vision and Pattern Recognition (CVPR) (2015)
11. Hartmann, W., Galliani, S., Havlena, M., Van Gool, L., Schindler, K.: Learned multi-patch similarity. International Conference on Computer Vision (ICCV) (2017)
12. Hirschmuller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2008)

13. Hirschmuller, H., Scharstein, D.: Evaluation of cost functions for stereo matching. Computer Vision and Pattern Recognition (CVPR) (2007)
14. Ji, M., Gall, J., Zheng, H., Liu, Y., Fang, L.: Surfacenet: An end-to-end 3d neural network for multiview stereopsis. International Conference on Computer Vision (ICCV) (2017)
15. Kar, A., Häne, C., Malik, J.: Learning a multi-view stereo machine. Advances in Neural Information Processing Systems (NIPS) (2017)
16. Kazhdan, M., Hoppe, H.: Screened poisson surface reconstruction. ACM Transactions on Graphics (TOG) (2013)
17. Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P.: End-to-end learning of geometry and context for deep stereo regression. Computer Vision and Pattern Recognition (CVPR) (2017)
18. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (TOG) (2017)
19. Knöbelreiter, P., Reinbacher, C., Shekhovtsov, A., Pock, T.: End-to-end training of hybrid cnn-crf models for stereo. Computer Vision and Pattern Recognition (CVPR) (2017)
20. Kutulakos, K.N., Seitz, S.M.: A theory of shape by space carving. International Journal of Computer Vision (IJCV) (2000)
21. Langguth, F., Sunkavalli, K., Hadap, S., Goesele, M.: Shading-aware multi-view stereo. European Conference on Computer Vision (ECCV) (2016)
22. Lhuillier, M., Quan, L.: A quasi-dense approach to surface reconstruction from uncalibrated images. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2005)
23. Luo, W., Schwing, A.G., Urtasun, R.: Efficient deep learning for stereo matching. Computer Vision and Pattern Recognition (CVPR) (2016)
24. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. Computer Vision and Pattern Recognition (CVPR) (2016)
25. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. Computer Vision and Pattern Recognition (CVPR) (2015)
26. Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J.M., Yang, R., Nistér, D., Pollefeys, M.: Real-time visibility-based fusion of depth maps. International Conference on Computer Vision (ICCV) (2007)
27. Moulon, P., Monasse, P., Marlet, R., Others: Openmvg. an open multiple view geometry library. `https://github.com/openMVG/openMVG`
28. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. IEEE International Symposium on Mixed and Augmented Reality (ISMAR) (2011)
29. OpenMVS: open multi-view stereo reconstruction library. `https://github.com/cdcseacave/openMVS`
30. Pix4D: `https://pix4d.com/`
31. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI) (2015)
32. Schönberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M.: Pixelwise view selection for unstructured multi-view stereo. European Conference on Computer Vision (ECCV) (2016)
33. Seitz, S.M., Dyer, C.R.: Photorealistic scene reconstruction by voxel coloring. International Journal of Computer Vision (IJCV) (1999)

34. Seki, A., Pollefeys, M.: Sgm-nets: Semi-global matching with neural networks. Computer Vision and Pattern Recognition Workshops (CVPRW) (2017)
35. Tola, E., Strecha, C., Fua, P.: Efficient large-scale multi-view stereo for ultra high-resolution image sets. Machine Vision and Applications (MVA) (2012)
36. Vu, H.H., Labatut, P., Pons, J.P., Keriven, R.: High accuracy and visibility-consistent dense multiview stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2012)
37. Xu, N., Price, B., Cohen, S., Huang, T.: Deep image matting. Computer Vision and Pattern Recognition (CVPR) (2017)
38. Yao, Y., Li, S., Zhu, S., Deng, H., Fang, T., Quan, L.: Relative camera refinement for accurate dense reconstruction. 3D Vision (3DV) (2017)
39. Zbontar, J., LeCun, Y.: Stereo matching by training a convolutional neural network to compare image patches. Journal of Machine Learning Research (JMLR) (2016)
40. Zhang, R., Li, S., Fang, T., Zhu, S., Quan, L.: Joint camera clustering and surface segmentation for large-scale multi-view stereo. International Conference on Computer Vision (ICCV) (2015)