

Efficient Multi-view Surface Refinement with Adaptive Resolution Control

Shiwei Li, Sing Yu Siu, Tian Fang*, Long Quan

Hong Kong University of Science and Technology
`{slibc,sysiuua,tianft,quan}@cse.ust.hk`

Abstract. The existing stereo refinement methods optimize a surface representation using a multi-view photo-consistency functional. Such optimization is iterative and requires repeated computation of gradients over all surface regions, which is the bottleneck affecting adversely the computational efficiency of the refinement. In this paper, we present a flexible and efficient framework for mesh surface refinement in multi-view stereo. The newly proposed Adaptive Resolution Control (ARC) evaluates an optimal trade-off between the geometry accuracy and the performance via curve analysis. Then, it classifies the regions into the significant and insignificant ones using a graph-cut optimization. After that, each region is subdivided and simplified accordingly in the remaining refinement process, producing a triangular mesh in adaptive resolutions. Consequently, the ARC accelerates the stereo refinement by severalfold by culling out most insignificant regions, while still maintaining a similar level of geometry details that the state-of-the-art methods could achieve. We have implemented the ARC and demonstrated intensively on both public benchmarks and private datasets, which all confirm the effectiveness and the robustness of the ARC.

1 Introduction

Recovering a realistic 3D model from images is the ultimate goal of Multiple View Stereo (MVS) methods. Boosted by the public MVS benchmarks [16, 15, 7], the accuracy of stereovision has dramatically increased in last decade. It is believed the key factor to high accuracy is the final surface refinement step. With a triangular mesh representing the surface, refinement is a process of iterative adjustment of vertex locations by optimizing multi-view photo-consistency.

Such iterative refinement is of heavy computation. The primary reason is the repeated computation of refinement gradient over all visible surface areas. Another reason is that *mesh subdivision* used in the refinement will dramatically increase the *#vertices* to be optimized. The higher density of mesh vertex also leads to slower mesh-related operations, *e.g.*, mesh smoothing, visibility testing.

According to our observation, not all regions of refinement contribute equally to the geometry improvement. For example, most planar or low-textured regions

* Tian Fang is the corresponding author

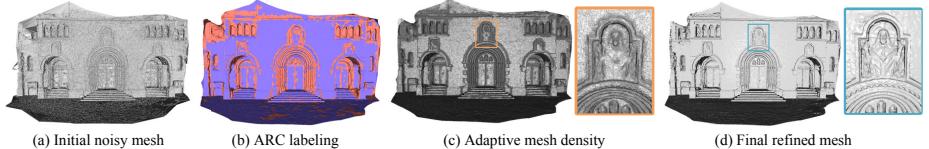


Fig. 1: With a noisy mesh as input (a), the ARC labels the mesh into two regions (b). Refinement applies only on the significant regions (orange), while the other insignificant regions (purple) will be culled out and simplified (c). This method greatly reduces the surface area to be optimized, but it is still able to produce valuable details (d).

barely have valuable refinement gradient, probably due to early convergence or lack of gradient on those regions. Refinement virtually produces no geometry improvement to them. Besides, mesh subdivision on such regions creates over-dense triangles, bringing extra computation and memory burden. In fact, these regions sometimes occupy quite a large proportion of the mesh surface (Fig. 4). Giving up their refinement can exchange for a decent performance speedup.

Unlike previous methods that target only at optimal photo-consistency, we also take the running time performance as our objective. To be specific, we quantify the performance and accuracy, and find an optimal trade-off in between which enables maximal performance speedup with minimal accuracy loss. Below, we demonstrate twofold contributions of our work.

Firstly, we present a mesh surface refinement framework with improvements to the baseline method [19]. Our refinement algorithm is divided into an *image registration* problem and a *gradient aggregation* problem. We employ a more efficient and direct approach to solve for the gradient of image similarity, which gives the steepest orientation for refinement. Besides, we identify the silhouette problem and handle it by explicitly culling out the problematic areas. The refinement framework is the fundamental that ensures a high accuracy reconstruction.

Secondly, we propose the novel *Adaptive Resolution Control* (ARC). The ARC labels the mesh into two regions (Fig. 1(b)), where the *active* regions are most contributive to geometry improvement, and the *inactive* regions are unimportant ones (usually planar or non-textured regions). To keep the labeling piecewise smooth, a graph cut optimization is employed. Only the *active* regions will be refined and subdivided, while the *inactive* regions will be discarded and simplified into fewer triangles. This leads to a mesh in adaptive resolution: the *active* regions have denser triangles while the *inactive* regions are sparser (Fig. 1(c)). Our method achieves a severalfold speedup thanks to the dramatic reduction of the refinement area and $\#\text{vertex}$ of the mesh. As shown in Fig. 1(d), our method is still able to preserve the fine details.

1.1 Related work

MVS starts with known camera parameters, aiming to reconstruct the dense representation of the target object. A huge volume of work has been conducted on MVS [15, 16]. Here, we only survey the works regarding surface refinement.

Surface refinement is the last step in MVS and the key factor to the final accuracy. Given a rough initial surface, it aims to refine the details by optimizing photo-consistency (usually minimizing the reprojection error).

Pons *et al.* [14] proposed a variational method [8] of surface refinement and scene-flow estimation for level set framework. Their formulation minimizes the global image reprojection error functional. Vu *et al.* [19] further extended their work to apply on discrete triangular meshes. Their method iteratively refines and subdivides the input triangular mesh, producing highly detailed results. Delaunoy *et al.* [5, 4] rigorously modeled the mesh refinement problem with the consideration of visibility change. Their formulation is further extended for the bundle adjustment problem [3]. Other than surface, patch-based methods [10, 12] apply the refinement to the patch representation (*i.e.*, normal and depth). Some earlier methods [9, 6, 18] estimated the refinement gradient using object silhouette information, but these methods are limited to conditioned scenarios.

Most refinement methods employ an iterative scheme to optimize the surface shape. Our refinement framework is closest to Vu’s method [19], which can be seen as the baseline of our method. In the rest of the paper, we first present an improved surface refinement framework in Section 2. Then we propose the novel *Adaptive Resolution Control* in Section 3. Intensive experiments have been conducted in Section 4 to support the effectiveness of the proposed method.

2 Mesh surface refinement

Previous refinement methods for triangular meshes produce impressive results [19, 3]. Our method sticks to this main rule, but we view it as a combination of two sub-problems (image registration and gradient aggregation). We also propose the fast photo-consistency (NCC) gradient computation (Section 2.2) and the silhouette culling (Section 2.3) as improvements to previous method.

2.1 The formulation

Denote a pair of images \mathbf{I}_i , \mathbf{I}_j and surface \mathbf{S} . As introduced by [14], the standard formulation minimizing their reprojection error is formulated as:

$$E_{i,j}(\mathbf{S}) = \int_{x_i \in \mathbf{I}_i \cap \mathbf{I}_j^{\mathbf{S},i}} -\mathcal{M}(\mathbf{I}_i(x_i), \mathbf{I}_j^{\mathbf{S},i}(x_i)) dx_i, \quad (1)$$

where $\mathbf{I}_j^{\mathbf{S},i}$ is the reprojection of image j in view i via surface \mathbf{S} , and \mathcal{M} is the image similarity measurement. $E_{i,j}(\mathbf{S})$ integrates the error over commonly visible area for image pair i and j . Then the error summing up all image pairs $E(\mathbf{S}) = \sum_{i,j} E_{i,j}(\mathbf{S})$ is minimized. Assuming camera parameters are correct and objects are Lambertian, the difference between \mathbf{I}_i and $\mathbf{I}_j^{\mathbf{S},i}$ is due to the inaccurate surface \mathbf{S} . Here, we separate the minimization into two sub-problems.

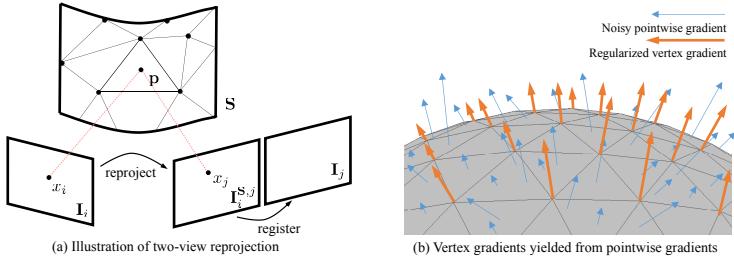


Fig. 2: (a) The two-view refinement problem is formulated as an image registration problem and a gradient aggregation problem. (b) The discrete vertex gradient is solved by a least square of pointwise gradient with regularization enforcement.

Image registration. The original formulation Eq. 1 measures the photo-consistency between \mathbf{I}_i and $\mathbf{I}_j^{\mathbf{S},i}$. Instead, we switch the measurement space to x_j coordinate, *i.e.*, measuring $\mathbf{I}_i^{\mathbf{S},j}$ and \mathbf{I}_j . This particular choice enables two sub-problems to be recombined via the proxy x_j . To maximize the image similarity \mathcal{M} , we take the partial derivative of $-\mathcal{M}$ to the coordinate x_j of its first argument:

$$\nabla(-\mathcal{M})[x_j] = -\frac{\partial_1 \mathcal{M}(\mathbf{I}_i^{\mathbf{S},j}, \mathbf{I}_j)(x_j)}{\partial x_j} = \mathbf{G}_{\mathbf{I}_i^{\mathbf{S},j}}(x_j) \in \mathbb{R}^2. \quad (2)$$

The 2D gradient field $\mathbf{G}_{\mathbf{I}_i^{\mathbf{S},j}}$ can be viewed as the optical-flow that registers $\mathbf{I}_i^{\mathbf{S},j}$ onto \mathbf{I}_j . We will show its fast computation in Section 2.2.

Gradient aggregation. We consider a two-view scenario (Fig. 2(a)): a surface point \mathbf{p} has two projected coordinates $x_i = \Pi_i(\mathbf{p})$, $x_j = \Pi_j(\mathbf{p})$. The image reprojection $\mathbf{I}_i^{\mathbf{S},j}$ deforms as surface \mathbf{S} deforms. As $\mathbf{G}_{\mathbf{I}_i^{\mathbf{S},j}}$ is the gradient optimizes \mathcal{M} , we solve for the surface gradient $\mathbf{G}_{\mathbf{S}}$ which induces the desired $\mathbf{G}_{\mathbf{I}_i^{\mathbf{S},j}}$. To bridge them, we replace Eq. 2 with the derivative to a surface variation $\delta\mathbf{S}$:

$$\begin{aligned} \nabla(-\mathcal{M})[\delta\mathbf{S}](x_i) &= -\left. \frac{\partial \mathcal{M}(\mathbf{I}_i^{\mathbf{S},j}(x_i \circ \Pi_{i,\mathbf{S}+\epsilon\delta\mathbf{S}}^{-1} \circ \Pi_j), \mathbf{I}_j)}{\partial \epsilon} \right|_{\epsilon=0} \\ &= \left(-\frac{\partial_1 \mathcal{M}(\mathbf{I}_i^{\mathbf{S},j}, \mathbf{I}_j)(x_j)}{\partial x_j} \right) \left(\frac{dx_j}{d\mathbf{p}} \right) \left(\frac{\partial \Pi_{i,\mathbf{S}+\epsilon\delta\mathbf{S}}^{-1}(x_i)}{\partial \epsilon} \right) \Big|_{\epsilon=0} \\ &= \left[\mathbf{G}_{\mathbf{I}_i^{\mathbf{S},j}}(x_j) \cdot \mathbf{J}_j \cdot \frac{\mathbf{d}_i}{\mathbf{N}^T \mathbf{d}_i} \right] \mathbf{N}^T \delta\mathbf{S}. \end{aligned}$$

Note that in line two, the first term $-\frac{\partial_1 \mathcal{M}(\mathbf{I}_i^{\mathbf{S},j}, \mathbf{I}_j)(x_j)}{\partial x_j} = \mathbf{G}_{\mathbf{I}_i^{\mathbf{S},j}}(x_j)$ is pre-computed. The second term $\frac{dx_j}{d\mathbf{p}} = \mathbf{J}_j$ is the Jacobian of projection matrix Π_j . The third term, assuming the surface movement is along its normal direction [14], can convert to $\frac{\partial \Pi_{i,\mathbf{S}+\epsilon\delta\mathbf{S}}^{-1}(x_i)}{\partial \epsilon} \Big|_{\epsilon=0} = \frac{\mathbf{N}^T \delta\mathbf{S}(\mathbf{p})}{\mathbf{N}^T \mathbf{d}_i} \mathbf{d}_i$, where \mathbf{N} is the normal of \mathbf{p} , and \mathbf{d}_i is the

joining vector from the camera center i to \mathbf{p} . Then, the gradient for a surface point \mathbf{p} is:

$$\mathbf{G}_S(\mathbf{p}) = \left[\mathbf{G}_{\mathbf{I}_i^{\mathbf{S},j}}(x_j) \cdot \mathbf{J}_j \cdot \frac{\mathbf{d}_i}{\mathbf{N}^T \mathbf{d}_i} \right] \mathbf{N}. \quad (3)$$

Regularized discretization. Here, an optimize-then-discretize strategy is employed. The surface is represented by triangular mesh $\mathbf{M} = \{\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_n\}$, and the vertex refinement gradient is denoted as \mathbf{G}_M . An arbitrary surface point \mathbf{p} can be written as the barycentric coordinate of the enclosing triangle vertices $\mathbf{p} = \sum_k \phi_k \mathbf{v}_k$, where $\sum_k \phi_k = 1$. This relation also holds for their gradient $\mathbf{G}_S(\mathbf{p}) = \sum_k \phi_k \mathbf{G}_M(\mathbf{v}_k)$. To solve for \mathbf{G}_M , we formulate it as a linear least square problem $\mathbf{A}_{[m \times n]} \mathbf{G}_M = \mathbf{G}_S$, where matrix \mathbf{A} fills with corresponding barycentric weights ϕ , and $m = \# \text{surface points}$, $n = \# \text{vertices}$ ($m \gg n$). As illustrated in Fig. 2(b), a pointwise gradient $\mathbf{G}_S(\mathbf{p})$ is sensitive to noise. However, the least-squared discrete gradient $\mathbf{G}_M(\mathbf{v})$ is much more regularized.

An additional regularization is applied to the data-term: the gradient of a vertex is expected to be smooth to its neighborhood: $\mathbf{G}_M(\mathbf{v}_i) = \frac{1}{N} \sum_{j \in N(i)} \mathbf{G}_M(\mathbf{v}_j)$. This relation for all the vertices can be written as $\beta \mathbf{B}_{[n \times n]} \mathbf{G}_M = 0$, where β is a weight adjusting the smoothness. Stacking up matrix \mathbf{A} and $\beta \mathbf{B}$ forms a massive sparse matrix, and the \mathbf{G}_M can be solved via bi-conjugate gradient method. The \mathbf{G}_M is applied to the mesh in each iteration: $\mathbf{M}_{i+1} = \mathbf{M}_i + \epsilon \mathbf{G}_M$.

Note that in previous method [19], the gradient of a vertex is the sum over its one-ring triangles from all pairs. Although it is faithful to its formulation, the gradient magnitude would be biased when the surface visibility is not balanced. *e.g.*, Regions viewed by more image pairs have larger magnitude. Our discretization based on a least square can prevent from the visibility bias problem.

Coarse-to-fine. To alleviate the local optimal problem, we adopt a coarse-to-fine strategy. Multiple scales of images are set up beforehand. The input mesh is first smoothed and simplified to a certain level, followed by the refinement by images from low-res to high-res gradually over the iterations. A triangle would be subdivided if its projection area covers more than 9 pixels in any image pair. The step size ϵ is globally adjusted according to the edge length of the mesh.

2.2 Fast NCC gradient

The image similarity gradient essentially drives the surface refinement. It is also the biggest performance bottleneck of the whole algorithm. Here, we provide a fast gradient computation on NCC similarity measurement.

In [5], the similarity metric is simply the rooted squared difference of pixel intensity $\|\mathbf{I}_i - \mathbf{I}_j\|_2$, which is fragile to inconsistent illumination. In [19, 14], they employ ZNCC as similarity metric, but the gradient $\frac{\partial_1 \mathcal{M}}{\partial \mathbf{x}}$ is separated into $\frac{\partial_1 \mathcal{M}}{\partial \mathbf{I}(x)} \frac{d\mathbf{I}(x)}{dx}$ using chain rules, where $\frac{d\mathbf{I}(x)}{dx} = \nabla \mathbf{I}(x)$ is simply the image gradient. We argue that it slows down the convergence due to two reasons: 1) as $\frac{\partial_1 \mathcal{M}}{\partial \mathbf{I}(x)}$ is a

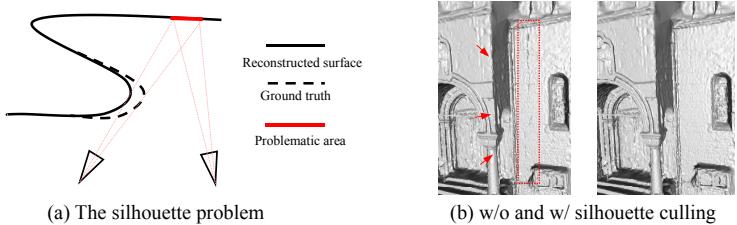


Fig. 3: Inaccurate surface induces image reprojection onto a wrong layer (a), and leads to a wrong refinement gradient ((b) left). With silhouette culling, this problem is avoided ((b) right).

scalar, it implicitly constrains the refinement gradient $\frac{\partial_1 \mathcal{M}}{\partial x}$ to be on the image gradient orientation $\nabla \mathbf{I}(x)$, but in fact it may not be the steepest orientation; 2) a single pixel intensity $\mathbf{I}(x)$ is used to connect the chain rule, but the real computation of ZNCC is over a neighborhood of x .

To improve, we resort to a more efficient and direct way to solve for $\frac{\partial_1 \mathcal{M}}{\partial x}$. Concretely, we use Normalized Cross Correlation (NCC) instead of the zero-mean version, which reduces the chance of zero denominator. Consider Eq. 2 as the computation for gradient that registers a dynamic image \mathbf{I}_d to a static image \mathbf{I}_s . We denote the scalar product $S(d, s, x) = \sum_{x_k \in N(x)} (\mathbf{I}_d(x_k) \mathbf{I}_s(x_k))$, and then $\text{NCC}(\mathbf{I}_d, \mathbf{I}_s)(x) = \frac{S(d, s, x)}{[S(d, d, x)S(s, s, x)]^{1/2}} = \frac{A}{B}$. The gradient is computed by taking the derivative to the coordinate x of dynamic image \mathbf{I}_d :

$$\mathbf{G}_{\mathbf{I}}(x) = \frac{\partial_1 \text{NCC}(\mathbf{I}_d, \mathbf{I}_s)(x)}{\partial x} = \frac{\partial}{\partial x} \left(\frac{A}{B} \right) = \frac{B \frac{\partial A}{\partial x} - A \frac{\partial B}{\partial x}}{B^2},$$

where

$$\begin{aligned} \frac{\partial A}{\partial x} &= \frac{\partial S(d, s, x)}{\partial x} = \sum_N (\mathbf{D}_d(x) \mathbf{I}_s(x)) = S(d', s, x), \\ \frac{\partial B}{\partial x} &= \frac{\partial [S(d, d, x)S(s, s, x)]^{1/2}}{\partial x} = \left[\frac{S(s, s, x)}{S(d, d, x)} \right]^{\frac{1}{2}} S(d', d, x). \end{aligned}$$

\mathbf{D} denotes the image gradient. The final formulation simplifies to:

$$\mathbf{G}_{\mathbf{I}}(x) = \frac{S(d', s, x) - S(d', d, x) \frac{S(d, s, x)}{S(d, d, x)}}{\sqrt{S(d, d, x)S(s, s, x)}}.$$

The computation of $\mathbf{G}_{\mathbf{I}}(x)$ is independent for every pixel x , making it perfectly suitable for GPU parallelism.

2.3 Silhouette culling

Due to the inaccurate initial mesh, the image i is potentially reprojected to a wrong depth layer. This often happens along the silhouettes of the object, as

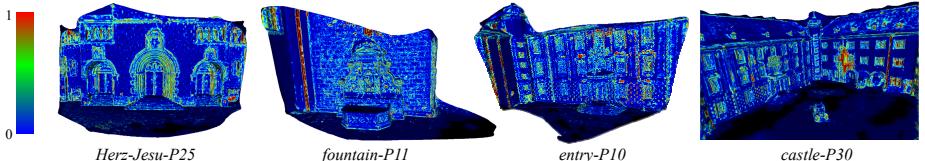


Fig. 4: The magnitude of refinement gradient for four EPFL dataset [16] in early iteration. Most regions such as flat walls or grounds have very small gradient values (blue). They have very little geometry changes before and after refinement.

shown in Fig. 3(a). While this problem is unsolved in the previous method, we handle it by explicitly detecting the silhouettes during the rendering for reprojection, and culling out problematic silhouette areas. A mesh edge \mathbf{E} is a silhouette edge w.r.t. view i if and only if its two incident triangles $t_{0/1}$ are front face and back face. *i.e.*, silhouette edges $\mathbf{SE} = \{\mathbf{E} | \langle \mathbf{N}_{t_0}, \mathbf{N}_{view} \rangle \oplus \langle \mathbf{N}_{t_1}, \mathbf{N}_{view} \rangle, t_0, t_1 \in N(\mathbf{E})\}$. Pixels on \mathbf{SE} are discarded in the refinement.

3 Adaptive resolution control

Motivated by the observation in Fig. 4, the ARC relaxes the original full refinement to partial refinement on selected regions. Specifically, the ARC segments the surface into two regions namely, *active* and *inactive*. *Active* represents those significant ones that will apply refinement. *Inactive* means those insignificant ones and would be discarded from refinement in exchange of performance gain.

Let f be a function that assigns each surface region R a label $f(R) \in \{\text{active}, \text{inactive}\}$. The trade-off can be formulated as an utility maximization:

$$u(f) = u_{accuracy}(f) + u_{time_reduction}(f). \quad (4)$$

$u_{accuracy}(f)$ measures the utility derived from accuracy of ARC refinement. This can be measured as the *geometry improvement* achieved by refining only *active* regions. $u_{time_reduction}(f)$ measures the utility derived from *time reduction* achieved by culling out *inactive* regions in refinement.

3.1 Quantification on triangular meshes

In the context of meshes, triangle is the smallest unit of surface region. We define two metrics on triangle to concretely formulate the trade-off problem on meshes.

Geometry improvement. As illustrated earlier, refinement on each vertex contributes differently to geometry improvement. To quantify the improvement, we borrow the *quadric error metric* used in mesh simplification [11], to capture the amount of geometry difference a vertex displacement can bring. This is a better alternative than vertex gradient magnitude alone because refinement has opposite goal¹ to simplification and hence should use the same set of metric. Let \mathbf{v}

¹ Simplification minimizes the geometry changes while refinement maximizes it.

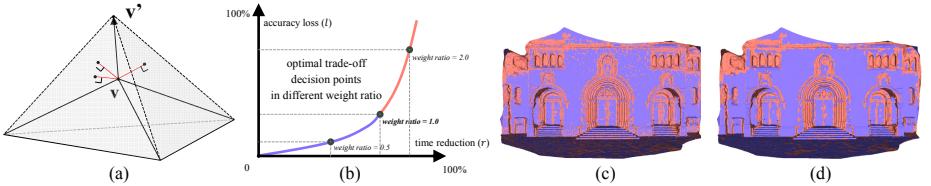


Fig. 5: (a) The *geometry improvement* of vertex is the maximum squared distance from \mathbf{v} to $\text{planes}(\mathbf{v}')$. (b) Trade-off curve between time reduction and accuracy loss. (c) Labeling by *optimal trade-off decision* $f^{optimality}$. (orange – *active*, purple – *inactive*). (d) Final labeling by graph-cut optimization.

and \mathbf{v}' be the same vertex before and after a refinement iteration. As shown in Fig. 5(a), we define *geometry improvement* (gi) for a vertex \mathbf{v} as the maximum of squared distances between \mathbf{v} and one-ring neighbor planes of \mathbf{v}' , referred as $\text{planes}(\mathbf{v}')$, and gi for a triangle as the average of its three vertices:

$$gi_{\mathbf{v}} = \max_{\mathbf{p} \in \text{planes}(\mathbf{v}')} \{(\mathbf{p}^t \mathbf{v})^2\}$$

$$gi_t = \frac{1}{3} \sum_i^3 gi_{\mathbf{v}_i},$$

where $\mathbf{v} = [v_x \ v_y \ v_z \ 1]^t$, $\mathbf{p} = [a \ b \ c \ d]^t$ represents a plane in standard form.

Running time cost. The majority of computation spent on the refinement gradient. The cost spent on a triangle is a factor of the number of visible image pairs times its area. Then, the *time cost* (tc) for a triangle t is formulated as:

$$tc_t = \frac{1}{2} |(\mathbf{v}_2 - \mathbf{v}_0) \times (\mathbf{v}_1 - \mathbf{v}_0)| \cdot (\# \text{visible image pair}(t)),$$

where $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$ are three vertices of t .

3.2 Optimal trade-off decision

Given the metrics introduced previously, we define the *cost effectiveness* (ce_t) of a triangle t as the ratio of its *geometry improvement* over its *time cost*, i.e., $ce_t = gi_t / tc_t$. A higher ce means more accuracy can be achieved over the same unit of *time cost* by labeling it as *active*. Therefore we should always label triangles as *inactive* from the lowest ce to the highest ce .

To better illustrate the effect of this labeling principle, we compute the ce_t for all triangles and sort them in ascending order. Then we obtain an accumulation curve by incremental summation of tc_t on x -axis, and gi_t on y -axis in this sorted order. Every point on this curve represents a labeling configuration based on the principle, which will label all triangles below and above that particular point as *inactive* and *active*. Then we normalize both axes to $[0, 1]$, and the x, y -axis can

be interpreted as the *time reduction* (r) = $\sum_{inactive} tc_{t_i} / \sum_{total} tc_{t_i}$ and *accuracy loss* (l) = $\sum_{inactive} git_{t_i} / \sum_{total} git_{t_i}$ (as shown in Fig. 5(b)).

This curve gives us flexibility to control the amount of trade-off. We can fix a threshold or range on either *time reduction* or *accuracy loss* according to our application needs. More importantly, we can transform the problem space from label assignment function f to 2D space $(r, l) \in curve$. We rewrite Eq. 4 as:

$$\begin{aligned} u(r, l) &= u_{accuracy}(l) + u_{time_reduction}(r) \\ &= w_l \cdot (1 - l) + w_r \cdot r, \end{aligned}$$

where w_l and w_r are the weights for *accuracy loss* and *time reduction*. The *optimal trade-off decision point* (r_o, l_o) is on the curve such that

$$u(r_o, l_o) = \max_{(r, l) \in curve} u(r, l),$$

which can be solved by taking derivative on $u(r, l)$. It can be deduced the optimal point (r_o, l_o) on curve has slope equal to $\frac{w_r}{w_l}$. This point represents the optimal labeling, *i.e.*, $f^{optimality}$ and it is unique because the slope of the curve is strictly increasing since it is already sorted. Note that *full refinement* can be seen as a special case represented by the point $(0, 0)$ on curve. The weight ratio $\frac{w_r}{w_l}$ is representing the relative importance of *time reduction* over *accuracy loss*. By default and in the following experiments, we use weight ratio = 1, which means putting equal weights on *accuracy loss* and *time reduction*.

3.3 Graph optimization

Labeling the mesh by the *optimal trade-off decision* alone maximizes our utility function, but it also makes the labeling fragmented into a lot of small regions. An example is given in Fig. 5(c). A desired labeling should be consistent with the data-term labeling while being piecewise smooth over the mesh. Therefore graph cut optimization [1] is employed to cope with this problem.

Let f be a labeling configuration that assigns each triangle t a label $f_t \in \{active, inactive\}$. The energy function of f formulates as the sum of three terms:

$$\mathbf{E}(f) = \mathbf{E}_{optimality}(f) + \mathbf{E}_{smoothness}(f) + \mathbf{E}_{prior}(f). \quad (5)$$

Optimality. It is desirable that the final labeling keeps as much fidelity as possible to the data-term label given by *optimal trade-off decision*, $f^{optimality}$. So $\mathbf{E}_{optimality}(f)$ accumulates the penalty for all the triangle labels that violate the optimality labels, *i.e.*, $\mathbf{E}_{optimality}(f) = \sum_i 1[f_{t_i}^{optimality} \neq f_{t_i}]$.

Smoothness. As a nature prior, the labeling should be piecewise smooth. More importantly, a smooth labeling enables an effective simplification applied to a larger pieces of the mesh. We simply use the Potts model $\mathbf{E}_{smoothness}(f) = \sum_{i,j \in e_{ij}} 1[f_{t_i} \neq f_{t_j}]$ to enforce the labeling smoothness between neighboring triangles t_i and t_j . From our experiences, we omit using a weighting scheme such as edge length $\|e_{i,j}\|$, because a further normalization will easily be affected by the longest edge. Instead, we use uniform weighting to achieve a reasonable balance with $\mathbf{E}_{optimality}(f)$.

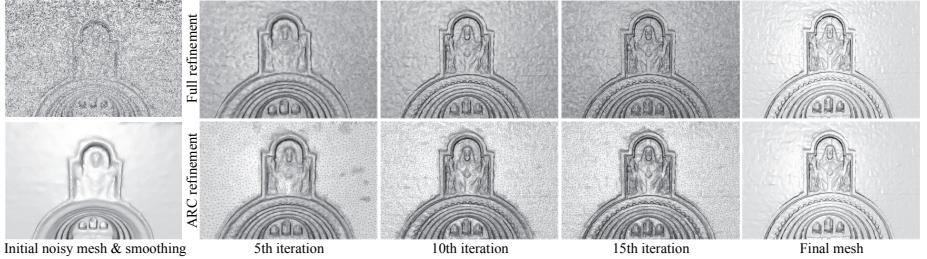


Fig. 6: Left: the noisy mesh is smoothed before refinement. Right: comparison between *full refinement* and *ARC refinement* over iterations. The *full refinement* generates an evenly dense mesh, while the *ARC refinement* produces a mesh in adaptive resolution: the valuable (*e.g.*, edges) region has much denser triangles than unimportant regions (*e.g.*, planes), but the final quality is very similar to the fully refined one.

Textureness prior. It is optional to add the textureness prior to the graph optimization. A sharp gradient change in 2D image does not always mean the real detail in 3D scene (*e.g.*, textured pattern on a flat wall), but it is true that, in most cases, a real 3D geometry detail will generate sharp gradient on its projected 2D image. We employ a prior energy to encourage the textured regions to be labeled as *active*. Specifically, we compute the average image gradient magnitude $\|\nabla \mathbf{I}(t)\|_2$ (normalized to $[0, 1]$) over the pixels on the image which has the largest projection area of the triangle t . *i.e.*, $\mathbf{E}_{prior}(f) = \sum_t \|\nabla \mathbf{I}(t)\|_2 \cdot [f_t = inactive]$.

Graph-cut optimization above yields a piecewise smooth labeling (as shown in Fig. 5(d)). Worth-mentioning, the labeling is naturally adapted to the scene. For example, the more non-textured regions the model has, the higher proportion will be labeled as *inactive*, and thus gives higher performance gain.

3.4 Combining with refinement

Recall that the refinement employs a coarse-to-fine strategy. By default an image pyramid of three levels and 20 iterations of refinement are used. The ARC labeling recomputes once the image level changes, so ARC only executes three times in the whole refinement, which is of trivial cost. The *active* triangles go through the refinement algorithm, and would be subdivided if necessary. The *inactive* ones undergo a QEM [11] simplification with a certain simplifying ratio. This ratio is set to 0.2 in our experiments. The dramatic drop on $\#triangle$ effectively accelerates the rendering and mesh operations as well. Then, these triangles are fixed in later iterations. Except for the visibility testing, all computations regarding to them are culled out.

Fig. 6 shows an evolutionary comparison between the baseline *full refinement* and our *ARC refinement*. The input noisy mesh will be smoothed before refinement. Overall, the *full refinement* produces much denser mesh, while the *ARC* method generates a very compact mesh in adaptive resolution. The final quality of both mesh surfaces are very close, and hardly be distinguished visually.

Table 1: Quantitative comparison on selected datasets of DTU benchmark [13]. The term “our full” – *full refinement*, and “our ARC” – *ARC refinement*. Smaller is better.

	scan 36				scan 63				scan 106			
	Accuracy		Completeness		Accuracy		Completeness		Accuracy		Completeness	
	Mean	Med.										
our full	0.2644	0.1677	0.9565	0.2319	0.8433	0.2267	0.6645	0.2663	0.2855	0.1822	0.9719	0.3083
our ARC	0.2646	0.1693	0.9602	0.2325	0.8516	0.2295	0.6687	0.2686	0.2865	0.1853	0.9744	0.3097
Vu [19]	0.2641	0.1683	0.9840	0.2337	0.8576	0.2266	0.6720	0.2567	0.2864	0.1850	0.9741	0.3040
tola [17]	0.3125	0.2007	1.0331	0.2856	0.9082	0.2711	0.7189	0.2985	0.3028	0.1902	0.9950	0.3256
furu [10]	0.6270	0.2778	0.6101	0.2930	2.3992	1.1192	0.6401	0.3849	0.7881	0.3028	0.7004	0.3244
camp [2]	0.5972	0.2317	0.4622	0.2317	2.4241	0.2782	0.4730	0.2782	0.5918	0.2793	0.6902	0.2793

4 Experiment

The proposed method is implemented and evaluated on a machine with 8-core Intel i7-4770K and 32GB of memory. The image reprojection and the refinement gradient is computed using OpenGL with a NVIDIA GTX980 graphic card.

In below experiments, two configurations of our method are compared. The *full refinement* refers to the highest accuracy refinement. The *ARC refinement* is with our *ARC* described in Section 3 using the default parameters.

4.1 Benchmarking

Our results are evaluated on two public MVS benchmarks [13, 15].

DTU benchmark [13] covers a wide range of objects and each consists of 49 or 64 different views of images at 1600×1200 resolution. The high accuracy camera calibration is provided along with the dataset. To test our refinement algorithm, we borrow the initial surface generated by the method *tola* [17], and evaluate the *accuracy* and *completeness* of our final refined mesh by following the author’s guideline. Note that the *accuracy* is defined as the distance from generated surface to the ground truth, and *completeness* the other way around.

We have tested the *full refinement* and the *ARC refinement* comparing to the baseline refinement method *Vu* [19] and three referencing methods provided in the benchmark, namely *tola* [17], *furu* [10] and *camp* [2]. Table 1 shows the statistics of three selected datasets (*scan 36*, *63*, *106*) each from a different category in the benchmark. All three refinement algorithms consistently improve the accuracy and completeness comparing to the initial mesh [17], and accuracy of our *full refinement* is the most competitive among all three datasets. Worth mentioning, our *ARC refinement* can achieve very close *accuracy* and *completeness* to *full refinement* by only refining partial regions.

Middlebury benchmark is the very first MVS benchmark developed by Seitz *et al.* [15]. Although the image resolution is relatively low (640×480) by today’s standard, it provides a fair platform for quantitative comparison (completeness and accuracy) with many other competing methods. Our method is not designed

Sort By	Temple Full 312 views		Temple Ring 47 views		Temple Sparse 16 views		Dino Full 363 views		Dino Ring 48 views		Dino Sparse 16 views	
	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp
Furukawa 3	0.49	99.6	0.47	99.6	0.63	99.3	0.33	99.8	0.28	99.8	0.37	99.2
Furukawa 2	0.54	99.3	0.55	99.1	0.62	99.2	0.32	99.9	0.33	99.6	0.42	99.2
Campbell	0.41	99.9	0.48	99.4	0.53	98.6						
ECCV2016_104	0.41	99.6	0.49	99.5	0.57	98.1	0.44	99.8	0.46	99.7	0.42	98.0
CVPR2016_466	0.41	99.7	0.5	99.5	0.69	97.8	0.26	99.8	0.25	99.9	0.34	99.7
DCV			0.73	98.2	0.66	97.3			0.28	100	0.3	100
Galliani	0.39	99.2	0.48	99.1	0.53	97.0	0.31	99.9	0.3	99.4	0.38	98.6
Vogiatzis2	0.5	98.4	0.64	99.2	0.69	96.9					0.51	98.7
Liu2					0.65	96.9						
SurfEvolution			0.56	98.9	0.78	96.8			0.56	97.7	0.66	97.6
Depth Fusion			0.53	99.5	0.72	96.8			0.46	99.5	0.42	97.8
CVPR2014_1287			0.51	99.1	0.7	96.6			0.51	98.7		

Fig. 7: Our results in Middlebury benchmark (highlighted in yellow).

for such explicit fore/back-ground objects, but we still submit our *full refinement* results to the benchmark challenge. As shown in Fig. 7, our results produce no less than 99.5% completeness on all full and ring datasets, and all items of the *temple* data rank within top 8th, which is very competitive among all methods. Overall, our *accuracy* is less competitive as our initial mesh tends to generate extra surface than the ground truth (*e.g.*, the bottom of the object).

4.2 Performance gain

We conducted experiments on public EPFL [16] and our private datasets to quantify the actual accuracy loss and performance gain of the *ARC refinement*.

To quantify the accuracy loss, we employ the *Hausdorff distance* to measure the difference between two meshes. The *accuracy loss* = $\frac{d_H(\mathbf{M}_{ARC}, \mathbf{M}_{full})}{d_H(\mathbf{M}_{initial}, \mathbf{M}_{full})}$, where \mathbf{M}_{full} is the fully refined mesh, \mathbf{M}_{ARC} the mesh by ARC refinement and $\mathbf{M}_{initial}$ the smoothed initial mesh, $d_H(\mathbf{M}_A, \mathbf{M}_B)$ means the distance from \mathbf{M}_A to \mathbf{M}_B . The measured processing time excludes irrelevant common operations such as I/O. The performance gain is simply the ratio of processing time.

As shown in Table 2, the ARC achieves a 3-6x performance gain among all eight datasets. The actual performance gain varies on each individual dataset. For example, the *castle-P30* enjoys the highest performance gain and the second lowest accuracy loss because of the large area of plain walls and grounds in that dataset. However, the *campus* dataset has the highest accuracy loss and worst performance gain. We believe this is caused by the large area of vegetation in the dataset, which is deformable and thus not suitable for refinement. Vertices at such area usually have large but incorrect gradient. After all, the accuracy loss is less than 10% for all datasets, which is tolerable for some applications.

We also record the *#vertex* at every refinement iteration for four EPFL datasets, shown in Fig. 8. The increase of the *#vertex* is caused by the subdivision on the mesh. The *#vertex* of *ARC refinement* keeps about one third of the *#vertex* of *full refinement*. The huge reduction in *#vertex* lowers the peak memory as well.

Table 2: The statistics of the performance comparison between *full refinement* and *ARC refinement*. Four EPFL datasets [16] and four large-scale datasets are evaluated.

dataset name	resolution	#image	full refinement #vertex	time (sec)	ARC refinement #vertex	time (sec)	accuracy loss	performance gain
Herz-Jesu-P25	3072 x 2048	25	2438K	318	663K	56	3.86%	5.68x
entry-P10	3072 x 2048	10	1564K	234	442K	43	7.28%	5.44x
castle-P30	3072 x 2048	30	2901K	327	687K	54	2.87%	6.06x
fountain-11	3072 x 2048	11	1455K	192	615K	49	3.74%	3.92x
Santa-Prisca	4000 x 3000	129	5819K	1534	2160K	407	6.40%	3.76x
memorial.hall	4000 x 3000	155	6928K	2395	2479K	438	2.17%	5.47x
Swanstone	4000 x 3000	217	9233K	2408	3536K	554	4.82%	4.35x
campus	6000 x 3376	276	35233K	7321	16513K	2420	9.68%	3.02x

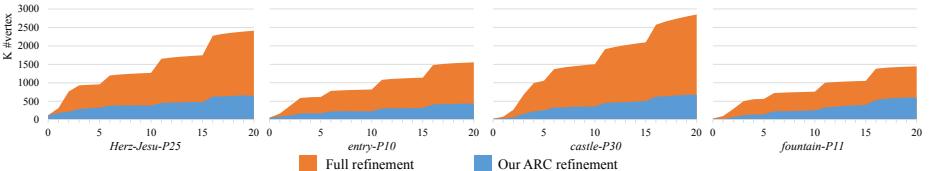


Fig. 8: Comparison of the *#vertex* throughout the iterations of refinement.

4.3 Qualitative evaluation

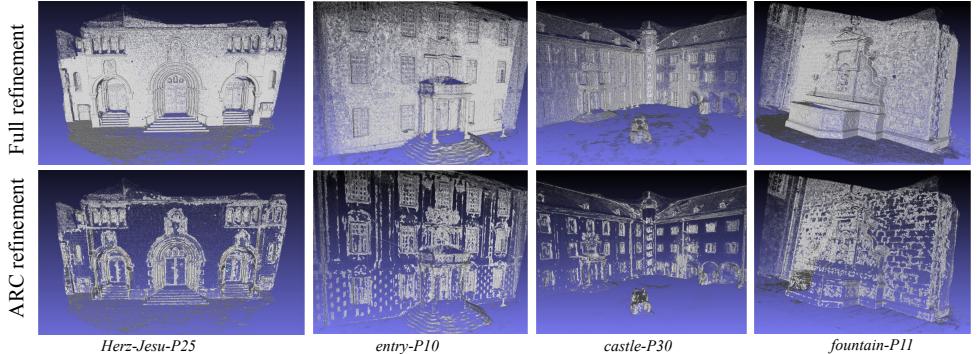
We show the qualitative comparison using EPFL dataset [16] in Fig. 9(a). Our *ARC refinement* produces adaptive vertex density over the triangular mesh, and overall, much lower number of vertices and triangles than the *full refinement*.

The proposed method can handle large-scale projects by employing a divide-and-conquer strategy. The huge mesh will be divided into a few pieces such that each single piece with its visible images can fit in the memory. As shown in Fig. 9(b), four private datasets below are all captured by UAV. The *Swanstone* dataset composes 217 images at 4K resolution. With a rough mesh surface as input, the *ARC refinement* is able to recover the fine details of the castle, such as the windows, or the crisp structures of the tower.

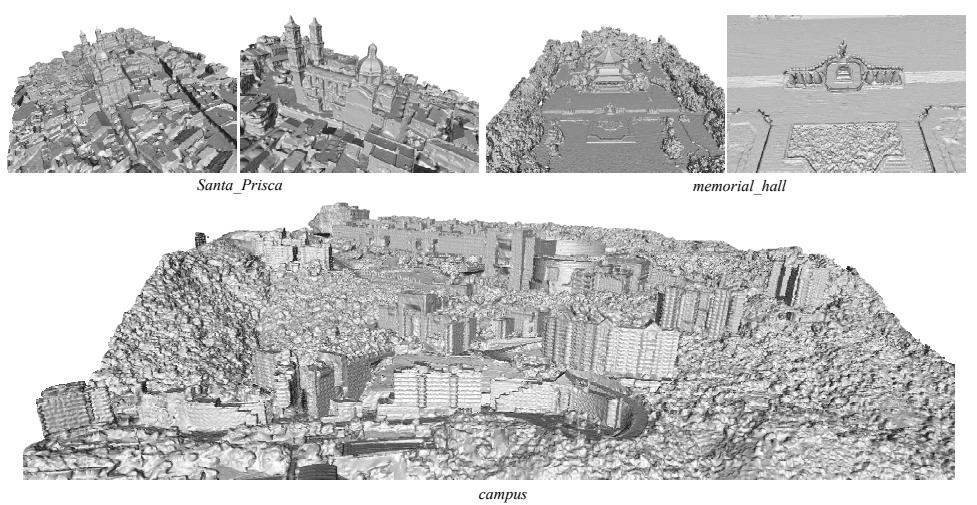
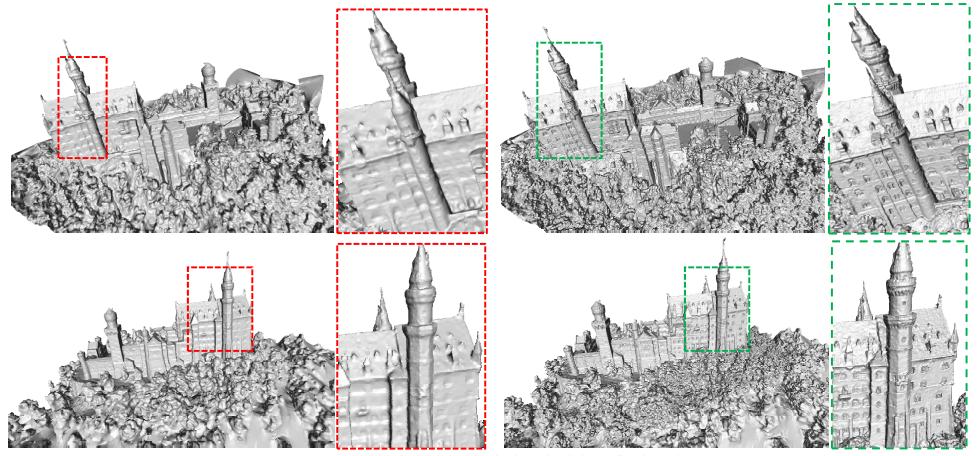
5 Conclusion

We have proposed the *ARC refinement* in this paper. The ARC estimates the most important regions to refinement and discard the other insignificant part in exchange for performance gain. The weight ratio controlling the trade-off between accuracy and performance is exposed and adjustable, which gives more flexibility to application demand. Our experiments demonstrate that ARC with default setting can achieve a dramatic speedup of 3-6x consistently with less than 10% accuracy loss comparing to baseline full refinement, which conveys the fact that refinement in most regions is indeed almost futile. This confirms the effectiveness and robustness of our ARC design.

Acknowledgements: This work is supported by Hong Kong RGC 16208614, T22-603/15N, Hong Kong ITC PSKL12EG02, and China 973 program, 2012CB316300. We thank Ximin Lyu, Lidong Ma and Hector Duran for the private dataset.



(a) Vertex density comparison between full and ARC refinement



(b) Qualitative results of large-scale datasets

Fig. 9: Qualitative evaluation of our ARC refinement. The upper four (EPFL benchmark [16]) compare the vertex density. The lower four are samples of the refined mesh surface of several large-scale projects. **Best viewed on screen.**

References

1. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via a graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(11), 1222–1239 (2001)
2. Campbell, N., Vogiatzis, G., Hernández, C., Cipolla, R.: Using multiple hypotheses to improve depth-maps for multi-view stereo. *Computer Vision–ECCV 2008* pp. 766–779 (2008)
3. Delaunoy, A., Pollefeys, M.: Photometric bundle adjustment for dense multi-view 3d modeling. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. pp. 1486–1493. IEEE (2014)
4. Delaunoy, A., Prados, E.: Gradient flows for optimizing triangular mesh-based surfaces: Applications to 3d reconstruction problems dealing with visibility. *International journal of computer vision* 95(2), 100–123 (2011)
5. Delaunoy, A., Prados, E., Piracés, P.G.I., Pons, J.P., Sturm, P.: Minimizing the multi-view stereo reprojection error for triangular surface meshes. In: *BMVC 2008–British Machine Vision Conference*. pp. 1–10. BMVA (2008)
6. Esteban, C.H., Schmitt, F.: Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding* 96(3), 367–392 (2004)
7. Fabri, A., Pion, S.: Cgal: The computational geometry algorithms library. In: *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. pp. 538–539. ACM (2009)
8. Faugeras, O., Keriven, R.: Variational Principles, Surface Evolution, PDE’s, Level Set Methods and the Stereo Problem. *Tech. Rep. RR-3021, INRIA* (Oct 1996)
9. Furukawa, Y., Ponce, J.: Carved visual hulls for image-based modeling. In: *Computer Vision–ECCV 2006*, pp. 564–577. Springer (2006)
10. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(8), 1362–1376 (2010)
11. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. pp. 209–216. ACM Press/Addison-Wesley Publishing Co. (1997)
12. Heise, P., Jensen, B., Klose, S., Knoll, A.: Variational patchmatch multiview reconstruction and refinement. In: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 882–890 (2015)
13. Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., Aanæs, H.: Large scale multi-view stereopsis evaluation. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. pp. 406–413. IEEE (2014)
14. Pons, J.P., Keriven, R., Faugeras, O.: Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision* 72(2), 179–193 (2007)
15. Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*. vol. 1, pp. 519–528. IEEE (2006)
16. Strecha, C., von Hansen, W., Gool, L.V., Fua, P., Thoennessen, U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. pp. 1–8. IEEE (2008)
17. Tola, E., Lepetit, V., Fua, P.: Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(5), 815–830 (2010)

18. Tylecek, R., Sara, R.: Refinement of surface mesh for accurate multi-view reconstruction. *The International Journal of Virtual Reality* 9(1), 45–54 (2010)
19. 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* 34(5), 889–901 (2012)