Massively Parallel Multiview Stereopsis by Surface Normal Diffusion

Silvano Galliani Katrin Lasinger Konrad Schindler

Photogrammetry and Remote Sensing, ETH Zurich

Abstract

We present a new, massively parallel method for high-quality multiview matching. Our work builds on the Patchmatch *idea: starting from randomly generated 3D planes in scene space, the best-fitting planes are iteratively propagated and refined to obtain a 3D depth and normal field per view, such that a robust photo-consistency measure over all images is maximized. Our main novelties are on the one hand to formulate Patchmatch in scene space, which makes it possible to aggregate image similarity across multiple views and obtain more accurate depth maps. And on the other hand a modified, diffusion-like propagation scheme that can be massively parallelized and delivers dense multiview correspondence over ten 1.9-Megapixel images in 3 seconds, on a consumer-grade GPU. Our method uses a slanted support window and thus has no fronto-parallel bias; it is completely local and parallel, such that computation time scales linearly with image size, and inversely proportional to the number of parallel threads. Furthermore, it has low memory footprint (four values per pixel, independent of the depth range). It therefore scales exceptionally well and can handle multiple large images at high depth resolution. Experiments on the DTU and Middlebury multiview datasets as well as oblique aerial images show that our method achieves very competitive results with high accuracy and completeness, across a range of different scenarios.*

1. Introduction

Reconstructing dense 3D shape from multiple images has been a topic of interest in computer vision for many years. Since camera pose estimation and multiview triangulation can be considered solved (at least for images that are suitable for subsequent dense reconstruction), the problem boils down to the fundamental task of image matching, *i.e.* establishing dense correspondence between images. The majority of the literature deals with the basic stereo setup with two images, *e.g.* [20, 23, 33, 31, 32]. It is evident that using more than two viewpoints will improve both the accuracy of the reconstructed 3D points (by triangulating from more rays) and the robustness against grossly wrong matches (by checking the coherence of redundant observations). Moreover, using more than two viewpoints alleviates the occlusion problem, and can reconstruct objects more completely, *e.g.* [7, 12]. On the other hand, the multiview setup exacerbates the problem that already many successful stereo methods do not scale up to realistic image sizes of several million pixels. Nevertheless, guided by the quality metrics used in standard benchmarks such as KITTI and Middlebury, most authors concentrate on accuracy and pay limited attention to scalability and runtime performance. Many existing algorithms become impractical when moving to larger sets of high-resolution images.

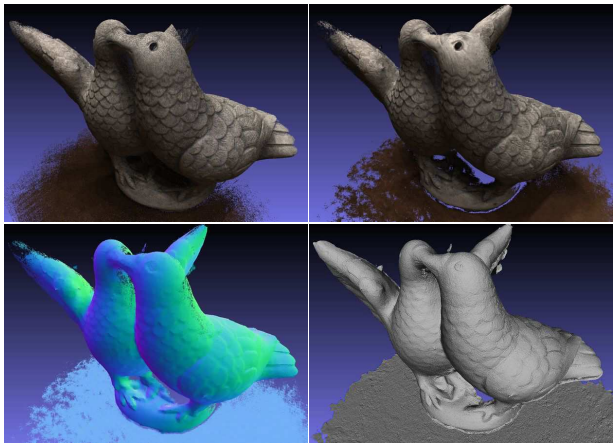


Figure 1: Results on one of the 80 evaluated objects on the DTU benchmark [22]. *Top left*: Ground truth point cloud; *top right*: reconstructed point cloud with texture; *bottom left*: color-coded surface normals; *bottom right*: reconstructed surface.

In this work we present a multiview matching method that delivers dense, accurate 3D point clouds while at the same time being efficient enough to handle large images. Our goal is a fast matcher which is nevertheless very accurate. On the recent DTU benchmark, our method reaches the best compromise between accuracy and completeness

(best accuracy with 2nd-best completeness, or best completeness with 2nd-best accuracy; see example in Fig. 1) still it can match ten 2-Megapixel images in less than 3 seconds on a standard desktop PC.

Local vs. global matching. Successful image matching has to strike a balance between photo-consistency of the corresponding image locations and regularity (typically piecewise smoothness) of the underlying surface.

Early models usually were *local*, meaning that the correspondence computation at a given location depends only on a local neighborhood. Local methods range from simple block matching to more sophisticated approaches that avoid a strong fronto-parallel bias, either by directly warping the image to a common plane [6, 10, 14], or using an oriented matching window that adapts to the surface geometry [5, 9]. Moreover, to avoid the characteristic fattening of foreground objects, it is common to adapt either the window shape [13, 23] or the weight of pixels within the window [38] at (putative) depth discontinuities.

Later research attempted to include the correlations induced by the smoothness prior in a more principled way, which leads to *global* methods that approximately maximize an objective defined over all pixels, usually via discrete labeling *e.g.* [11, 18, 27] or variational inference [31].

Nowadays photographic images routinely have on the order of 10 million pixels. Therefore there is a need for matching algorithms whose complexity is low – ideally at most linear in the number of pixels. At the same time, the large image dimensions also call for algorithms that are memory-efficient, especially in the multiview case, where evidence of multiple images is exploited to create the correct match. Consequently, there has been a renewed interest for local matching algorithms. In spite of their simplicity, modern local matchers [5, 12, 32] are accurate enough to compete with their global counterparts, as demonstrated for example by the DTU [22] and KITTI [15] benchmarks.

Local multiview methods. In their seminal work, Okutomi and Kanade [30] accumulate Sum of Squared Difference (SSD) cost values from different stereo pairs in a set of multiple images, and select the depth with the lowest cumulative cost. The plane-sweeping method [8] is an early example of true multiview matching. Evidence from multiple images is accumulated on a plane that moves through the scene space along its normal. For every cell on the plane the position with the highest support is chosen. More recently Gallup *et al.* [14] have proposed to align the plane to the dominant orientation in the scene. Hu and Mordohai [19] also start from plane-sweeping, and carefully propagate the uncertainty in order to exploit it during the subsequent fusion of multiple depth maps.

Furukawa and Ponce [12] relax the requirement to find a correspondence for every single pixel. Instead, they start from sparse, reliable seed points and iteratively grow the set of point matches from there, to obtain a quasi-dense point cloud. The method introduces several heuristic filters and delivers quite impressive results. Tola *et al.* [37] directly address the problem of high resolution image sets by matching a fast descriptor between pairs of images over the epipolar line and reconstructing only points with unique response. Campbell *et al.* [7] explicitly address the problem of ambiguous matching by considering multiple depths per point and including an unknown state in their MRF optimization.

Points vs. surfaces. Multi-view stereo methods can be classified according to which representation they are based on, following the taxonomy of Seitz et al. [34]. In particular, the 3D scene can be represented by voxels, level-sets, polygon meshes, or depth maps. In this context it should be emphasized that depth maps are still a point-wise representation – triangulating every pixel in a depth map leads to a 3D point cloud, similar to those generated with RGBD sensors or laser scanners. On the contrary, the three other representations all must solve (at least implicitly) the additional step from the point cloud to the underlying surface. This may be useful for many applications, but is a considerably harder and less well-defined task. Moreover, some application domains like industrial metrology or surveying in fact prefer 3D point clouds as a primary product. In our work we mainly aim to recover depth maps, respectively 3D point clouds. We see surface fitting as a subsequent step that is largely independent of the matching – in fact the most popular approaches [21, 25, 28] are rather agnostic about the preceding matcher, and we found the widely used Poisson method [25] to work well for our point clouds.

Exhaustive vs. randomized search. Typically, matching algorithms require a large amount of memory, because they keep track of the cost associated with *every* possible disparity value, in order to select the most suitable one, *e.g.* [11, 18, 23, 32]. Note that for a fixed depth range the number of observable disparities grows linearly with the image resolution, too. A recent exception from the strategy of "comparing all possible disparities" is *PatchMatch Stereo* [5]. That method adopts a randomized, iterative algorithm for approximate patch matching [3], which allows one to quickly find a good solution within a vast search space without having to browse through all possibilities. The resulting low memory requirements (independent of the disparity range) make *Patchmatch Stereo* well-suited for large images or memory-constrained environments, including implementation on GPU which modify the original sequential propagation scheme [1, 2, 17, 40]. Zheng *et al.* [40] employ the Patchmatch propagation scheme for multiview reconstruction, but without considering slanted surfaces. Their focus lies on view selection when aggregating evidence over multiple cameras. A probabilistic graphi-

cal model serves to jointly address view selection and depth estimation. To the best of our knowledge, [35] is the only other work that runs Patchmatch Stereo in scene space, for only pairwise stereo matching.

Contribution. We present ***Gipuma***, a simple, yet powerful multiview variant of Patchmatch Stereo with a new, highly parallel propagation scheme.

Our first contribution addresses computational efficiency: standard Patchmatch is sequential in nature, since it propagates information diagonally across the image pixel-by-pixel. A little parallelisation can be achieved by procedures such as aligning the propagation direction with the image axes and running rows/columns in parallel [1, 2, 17, 40], but these still do not fully harness the capabilities of current hardware. Instead, we propose a new diffusion-like scheme that operates on half of all pixels in an image in parallel with a red-black (checkerboard) scheme. It turns out that this arguably more local propagation, which is particularly suitable for modern many-core GPUs, works as well as the standard Patchmatch procedure, while being a lot faster.

The second contribution aims for accuracy and robustness: we extend PatchMatch Stereo from a two-view to a multiview matcher to better exploit the redundancy in multi-view datasets. The Patchmatch Stereo method by construction recovers also a normal in disparity space at every pixel. The starting point for our extension is the observation that one can just as well define the normals in Euclidean 3D scene space. In that case they immediately define a local tangent plane at every surface point, and thus an associated homography (respectively, a pair of slanted support windows) between any two images viewing the surface. The explicit estimation of the surface normal makes it possible to utilize plane-induced homographies when checking photo-consistency between different views. It avoids epipolar rectification and allows one to aggregate evidence over multiple images in generic configuration.

The described multiview setup still needs a reference image to fix the parametrization of the surface. Hence, we first compute depth using every image in turn as reference, and then fuse the results into one consistent 3D reconstruction. However, we prefer to carefully exploit the multiview information at the level of photo-consistency, and then use a rather basic fusion scheme to merge them into a consistent 3D point cloud. This is in contrast to some other methods that start from efficiently computable, but noisy depth maps and merge them with sophisticated fusion algorithms, which (at least implicitly) have to solve the additional problem of surface fitting [21, 28, 39].

We will show in our experiments that our implementation yields state-of-the-art multiview reconstruction on a variety of datasets.

2. Patchmatch Stereo

We start by briefly reviewing the original *Patchmatch Stereo* method [5], to set the scene for our extensions.

Patchmatch for rectified stereo images. The core of Patchmatch stereo is an iterative, randomized algorithm to find, for every pixel $\mathbf{p}$, a plane $\pi_{\mathbf{p}}$ in disparity space such that the matching cost m in its local neighborhood is minimized. The cost at pixel $\mathbf{p}$ is given by a dissimilarity measure ρ, accumulated over an adaptive weight window $W_{\mathbf{p}}$ around the pixel. Let $\mathbf{q}$ denote the pixels in the reference image that fall within the window, and let $\pi_{\mathbf{p}}$ be a plane that brings each pixel $\mathbf{q}$ in correspondence with a pixel location $\mathbf{q}'_{\pi_{\mathbf{p}}}$ in the other image. Then the matching cost is

$$m(\mathbf{p}, \pi_{\mathbf{p}}) = \sum_{\mathbf{q} \in W_{\mathbf{p}}} w(\mathbf{p}, \mathbf{q})\, \rho(\mathbf{q}, \mathbf{q}'_{\pi_{\mathbf{p}}}). \tag{1}$$

The weight function $w(\mathbf{p}, \mathbf{q}) = e^{-\frac{\|I_{\mathbf{p}} - I_{\mathbf{q}}\|}{\gamma}}$ can be seen as a soft segmentation, which decreases the influence of pixels that differ a lot from the central one. We use a fixed setting $\gamma = 10$ in all experiments.

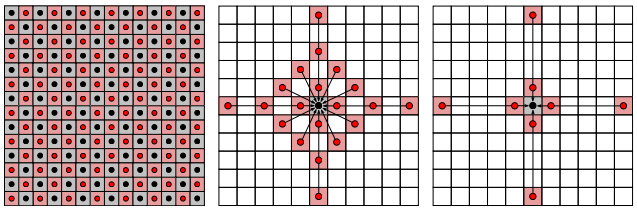
The cost function ρ consists of a weighted combination of absolute color differences and differences in gradient magnitude. More formally, for pixels $\mathbf{q}$ and $\mathbf{q}'_{\pi_{\mathbf{p}}}$ with colors $I_{\mathbf{q}}$ and $I_{\mathbf{q}'_{\pi_{\mathbf{p}}}}$

$$\begin{aligned}\rho(\mathbf{q}, \mathbf{q}'_{\pi_{\mathbf{p}}}) = (1-\alpha) \cdot \min(\|I_{\mathbf{q}} - I_{\mathbf{q}'_{\pi_{\mathbf{p}}}}\|, \tau_{col}) \\ + \alpha \cdot \min(\|\nabla I_{\mathbf{q}} - \nabla I_{\mathbf{q}'_{\pi_{\mathbf{p}}}}\|, \tau_{grad})\ ,\end{aligned} \tag{2}$$

where α balances the contribution of the two terms and τ_{col} and τ_{grad} are truncation thresholds to robustify the cost against outliers. In all our experiments we set $\alpha = 0.9$, $\tau_{col} = 10$ and $\tau_{grad} = 2$.

Sequential propagation. The Patchmatch solver initializes the plane parameters (disparity and normal) with random values. It then sequentially loops through all pixels of the image, starting at the top left corner. Good planes are propagated to the lower and right neighbors, replacing the previous values if they reduce the cost over the slanted support window. Additionally, it is proposed to also propagate planes between the two views. The propagation is interleaved with a refinement of the plane parameters (using bisection). After finishing a pass through all pixels of the image, the entire process is iterated with reversed propagation direction. Empirically, 2-3 iterations are sufficient. For optimal results the disparity image is cleaned up by *(i)* removing pixels whose disparity values are inconsistent between the two views; *(ii)* filling holes by extending nearby planes; and *(iii)* weighted median filtering.

Plane parameterization. In Patchmatch stereo, the $\pi_{\mathbf{p}}$ are planes in disparity space, *i.e.* 3D points $\mathbf{P}$ =



(a) (b) (c)

Figure 2: The propagation scheme: (a) Depth and normal are updated in parallel for all red pixels, using black pixels as candidates, and vice versa. (b) Planes from a local neighborhood (red points) serve as candidates to update a given pixel (black). (c) Modified scheme for speed setting, using only inner and outermost pixels of the pattern.

$[x, y, disp]^\top$ must fulfill the plane equation

$$\tilde{\mathbf{n}}^\top \mathbf{P} = -\tilde{d} \quad , \quad disp = -\frac{1}{\tilde{n}_z}(\tilde{d} + \tilde{n}_x x + \tilde{n}_y y) \quad , \quad (3)$$

with normal vector $\tilde{\mathbf{n}}$ and distance $\tilde{d}$ to the origin. This definition yields an affine distortion of the support windows in the rectified setup [17].

3. Red-Black Patchmatch

3.1. Surface normal diffusion

The standard Patchmatch procedure is to propagate information diagonally across the image, alternating between a pass from top left to bottom right and a pass in the opposite direction. The algorithm is sequential in nature, because every point is dependent on the previous one. Although several authors have proposed a parallel propagation scheme [1, 2, 17, 40], all of them still inherited from the original Patchmatch that one propagates sequentially across the whole image.

Instead, we propose a new *diffusion*-like scheme specifically tailored to multi-core architectures such as GPU processors. We partition the pixels into a "red" and "black" group in a checkerboard pattern, and simultaneously update all black and all red ones in turn. Possible candidates for the update at a given pixel are only points in a local neighborhood that belong to the respective other (red/black) group, see Fig. 2a.

The red-black (RB) scheme is a standard trick to parallelize message-passing type updating schemes, *c.f.* the red-black Gauss-Seidel method for linear equation solving. Red-black acceleration has also been proposed for Belief Propagation [11]. In fact Patchmatch can be interpreted as a form of Belief Propagation in the continuous space [4]. In contrast to these applications of the RB-scheme we look beyond the immediate neighbors. Our standard pattern uses 20 local neighbors for propagation, Fig. 2b. Thanks to the larger neighborhood we converge to a good solution already with a low number of iterations, see Fig. 3. The depicted scheme turned out to be a good compromise between the cost for each propagation step and the number of iterations needed to diffuse the information far enough. The number of iterations is fixed to 8 in all our experiments. At this point the depth map has practically converged and changes only marginally.

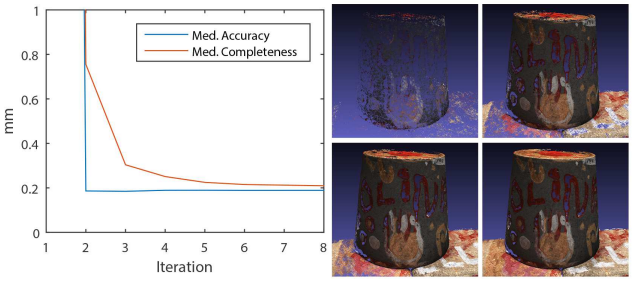


Figure 3: *Left:* Accuracy and completeness for increasing number of iterations for the object visualized on the right. *Right:* Reconstruction after iteration 2, 3, 4 and 8.

3.2. Sparse matching cost

We use a similar matching cost as proposed in the original Patchmatch paper [5]. The only difference is that we consider only intensity rather than color differences. The performance improvement when using RGB is tiny and in our view does not justify a threefold increase in runtime. To further speed up the computation we follow the idea of the so-called Sparse Census Transform [41] and use only every other row and column in the window when evaluating the matching cost, resulting in a $4\times$ gain. Empirically, we do not observe any decrease in matching accuracy with this sparse cost.

The method is particularly useful for Patchmatch-type methods. Such methods require larger window sizes, because compared to the disparity a larger neighborhood is needed to reliably estimate the normal. Depending on the image scale, the necessary window size is typically at least 11x11 pixels, but can reach up to 25x25 pixels.

3.3. Implementation details

We have implemented ***Gipuma*** in CUDA, and tested it on recent gaming video cards for desktop computers. For our experiments we use the Nvidia GTX 980. Images are mapped to texture memory, which provides hardware-accelerated bilinear interpolation to warp the support window between views. To limit the latency when reading from GPU memory we make extensive use of *shared memory* and cache the support window of the reference camera. We release our code as open-source software under the GPLv3 license.

Runtime. The runtime of our method is influenced mainly by three factors: the number of images considered for matching, the image resolution, and the size of the matching window (which in practice is roughly proportional to the image size).

For images of resolution 1600×1200 the runtime to generate a single depthmap with 10 images and window size of 25 is 50 seconds, when using our fast setting as described in Sec. 5.1 and windows size 15 the runtime for the same number of images is 2.7 seconds.

To generate a Middlebury depthmap from 10 views with a resolution of 640×480 the runtime is 2.5 seconds.

4. Multi-view Extension

4.1. Parameterization in scene space

Disparity, by definition, is specific to a pair of rectified images. Instead, we propose to operate with planar patches in Euclidean scene space. This variant has several advantages. First, it avoids epipolar rectification, respectively explicit tracing of epipolar lines, which is a rather unnatural and awkward procedure in the multiview setup. Second, it delivers, as a by-product, a dense field of surface normals in 3D scene space. This can be used to improve the subsequent point cloud fusion (*e.g.* one can filter pixels with consistent depth but inconsistent normal) as well as directly provide the necessary normal used for surface reconstruction [26]. Then, it allows the data cost to directly aggregate evidence from multiple views: the cost per-pixel is computed by considering the cost of the reference camera with respect to all the other selected views.

Finally, the modification comes at little extra cost: the mapping between any two images is a plane-induced homography [16], corresponding to a 3D matrix-vector multiplication, see Fig. 4.

In the Euclidean scene-space the plane equation $\mathbf{n}^\top \mathbf{X} = -d$ holds for 3D object points $\mathbf{X} = [X, Y, Z]^\top$. Finding the object point amounts to intersecting the viewing ray with the plane in space. W.l.o.g. one can place the reference camera at the coordinate origin. With the intrinsic calibration matrix K, the depth at a pixel $\mathbf{x} = [x, y]^\top = [\mathsf{K}|\mathbf{0}]\mathbf{X}$ is then related to the plane parameters by

$$Z = \frac{-dc}{[x-u,\ \alpha(y-v),\ c] \cdot \mathbf{n}} \quad , \quad \mathsf{K} = \begin{bmatrix} c & 0 & u \\ 0 & c/\alpha & v \\ 0 & 0 & 1 \end{bmatrix} . \quad (4)$$

where u, v is the principal point in pixels and c, $\frac{c}{\alpha}$ represent the focal length of the camera in pixels.

The image point $\mathbf{x}$ in the reference camera $\mathsf{K}[\mathsf{I}|\mathbf{0}]$ is then related to the corresponding point $\mathbf{x}'$ in a different camera $\mathsf{K}'[\mathsf{R}|\mathbf{t}]$ via the plane-induced homography

$$\mathsf{H}_\pi = \mathsf{K}'(\mathsf{R} - \frac{1}{d}\mathbf{t}\mathbf{n}^\top)\mathsf{K}^{-1} \quad , \quad \mathbf{x}' = \mathsf{H}_\pi \mathbf{x} \, . \quad (5)$$

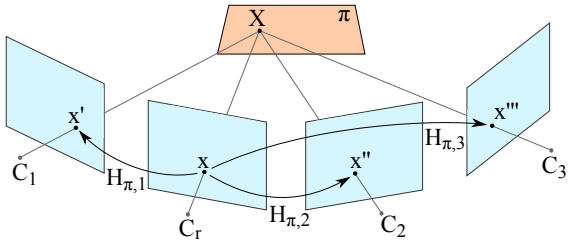


Figure 4: Multi-view setup with four cameras and homographies from reference camera C_r to three other cameras.

Initialization. When operating in scene space, one has to take some care to ensure a correct, unbiased random initialization of the Patchmatch solver. To efficiently generate random normals that are uniformly distributed over the visible hemisphere we follow [29]. Two values q_1 and q_2 are picked from a uniform distribution in the interval $(-1, 1)$, until the two values satisfy $S = q_1^2 + q_2^2 < 1$. The mapping

$$\mathbf{n} = \left[1 - 2S \ , \ 2q_1\sqrt{1-S} \ , \ 2q_2\sqrt{1-S}\right]^\top \quad (6)$$

then yields unit vectors equally distributed over the sphere. If the projection $[u, v, c]^\top \mathbf{n}$ onto the principal ray is positive, the vector $\mathbf{n}$ is inverted.

Furthermore, one should account for the well-known fact that the depth resolution is anisotropic: even if the matching is parametrized in scene space, the similarity is nevertheless measured in image space. It follows that the measurement accuracy is approximately constant over the disparity range, respectively inversely proportional to the depth. Therefore it is advisable to uniformly draw samples from the range of possible disparities and convert them to depth values (*i.e.* supply a more densely sampled set of depths to chose from in the near field, where they make a difference; and a sparser set in the far field, where small variations do not produce an observable difference). For the same reason, the search interval for the plane refinement step should be set proportional to the depth.

4.2. Cost computation over multiple images

When using multiple views, the question arises how to best combine the pairwise dissimilarities between images into a unified cost. In our implementation, we only consider the pairwise similarities between the reference image and all other overlapping views, but not those between pairs of non-reference images.

View selection For a given reference image, we first exclude all views whose viewing directions differ from the reference image by less than α_{min} or by more than α_{max}. The two thresholds correspond to the empirical observation that baselines $< \alpha_{max}$ are too small for triangulation and lead to overly high depth uncertainty, whereas baselines $> \alpha_{max}$

have too big perspective distortions to reliably compare appearance [37]. The selection of α_{min} and α_{max} is dataset dependent.

In big datasets where the angle criteria still produces too many views, we propose to randomly pick a subset S of views within this selection only if the runtime performance is preferred over accuracy, see Sec. 5. When used, we set $S = 9$.

Cost aggregation For a specific plane π, we obtain a cost value m_i from each of the N comparisons. There are different strategies how to fuse these into a single multiview matching cost.

One possible approach is to accumulate over all n cost values, as proposed by Okutomi and Kanade [30]. However, if objects are occluded in some of the views, these views will return a high cost value even for the correct plane, and thereby blur the objective. In order to robustly handle such cases we follow Kang et al. [24]. They propose to include only the best 50% of all N cost values, assuming that at least half of the images should be valid for a given point. We slightly change this and instead of the fixed 50% introduce a parameter K, which specifies the number of individual cost values to be considered,

$$m_{srt} = \text{sort}_{\uparrow}(m_1 \dots m_N) \quad , \quad m_{mv} = \sum_{i=1}^{K} m_i \; . \qquad (7)$$

The choice of K depends on different factors: in general, a higher value will increase the redundancy and improve the accuracy of the 3D point, but also the risk of including mismatches and thereby compromising the robustness. Empirically, rather low values tend to work better, in our experiments we use $K = 3$ or less for very sparse datasets.

4.3. Fusion

Like other multiview reconstruction schemes, we first compute a depth map for each view by consecutively treating all N views as the reference view. Then, the N depth maps are fused into a single point cloud, in order to eliminate wrong depth values and to reduce noise by averaging over consistent depth and normal estimates. Our approach follows the philosophy to generate the best possible individual depth maps, and then merge them into a complete point cloud in a straightforward manner.

Consistency Check Mismatches occur mainly in textureless regions and at occlusions, including regions outside of a camera's viewing frustum. Many such cases can be detected, because the depths estimated w.r.t. different viewpoints are not consistent with each other. To detect them, we again declare each image in turn the reference view, convert its depth map to a dense set of 3D points and reproject them to each of the $N - 1$ other views, resulting in a 2D coordinate $\mathbf{p}_i$ and a disparity value $\hat{d}_i$ per view. A match

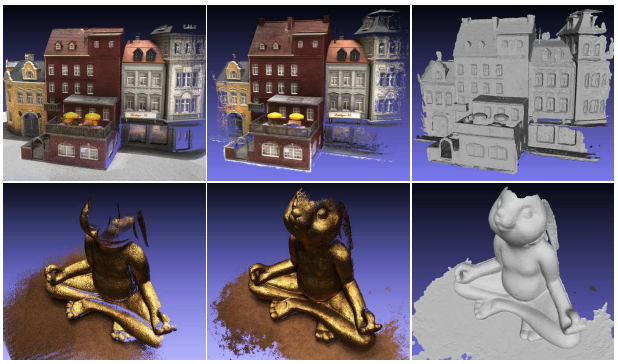


Figure 5: Reconstruction results of two DTU objects. From left to right: ground truth point cloud, textured point cloud and triangulated mesh surface.

is considered consistent if $\hat{d}_i$ is equal to the disparity value d_i stored in the corresponding depth map, up to a tolerance of f_ϵ pixels. The threshold depends on the scale of the reconstructed scene. We further exploit the estimated surface normals and also check that the normals differ by at most f_{ang}, in our experiments set to 30°. If the depth in at least f_{con} other views is consistent with the reference view, the corresponding pixel is accepted. Otherwise, it is removed. For all accepted points the 3D position and normal are averaged directly in scene space over all consistent views to suppress noise.

Accuracy vs. completeness The fusion parameters f_ϵ , f_{ang} and f_{con} filter out 3D points that are deemed unreliable, and thus balance accuracy against completeness of the multiview reconstruction. Different applications require a different trade-off (*e.g.*, computer graphics applications often prefer complete models, whereas in industrial metrology sparser, but highly accurate reconstructions are needed). We explore different setting in our experiments, see Sec. 5. Note that the fusion step is very fast ($\approx$ 15 seconds for 49 depthmaps of size 1600×1200) and does not change the depthmaps. One can thus easily switch from a more accurate to a more complete reconstruction, or even explore different levels interactively.

5. Results

We evaluate our multiview stereo GPU implementation on different datasets. We start with quantitative results on the recent DTU dataset for large scale multiview stereo [22]. To put our method in context we also evaluate on the Middlebury multiview benchmark [34], although the images in the dataset are very small by today's standards, and performance levels have saturated. When a triangulated mesh is required, we directly use our point cloud and normals with *Screened Poisson* reconstruction [26] with the program pro-

Figure 6: Reconstruction results for our three different settings. From left to right: *ours*, *ours comp*, *ours fast*. Note how the complete version is able to close the holes around the eye but suffers from boundary artifacts along the crest. On the other hand, the fast version, similar to the original, presents bigger holes around the eye and on the right side of the mantle.

vided by the authors.

Additional qualitative results on aerial images are shown in Sec. 5.3 to demonstrate the broad applicability of our method.

5.1. DTU Robot Image Dataset

As our main testbed, we use the recent DTU large scale multiview dataset [22]. It contains 80 different objects, each covered by 49–64 images of resolution 1600×1200 pixels. The captured scenes have varying reflectance, texture and geometric properties and include fabric, print, groceries, fruit and metallic sculptures, see Fig. 5. The images have been captured with different lighting conditions, and with two different distances to the object, using a robot arm to accurately position the cameras. We use only the most diffuse lighting to select the same set of images as used by the other methods. The ground truth has been acquired with a structured light scanner.

We followed the protocol specified by the authors of the dataset, *i.e.* we compute the mean and median reconstruction errors, both for the estimated 3D point cloud and for a triangulated mesh derived from the points. Accuracy is defined as the distance from the surface to the ground truth, and completeness from the ground truth to the surface. In this way completeness is expressed in mm and not as a percentage.

Compared to other methods, we achieve the highest accuracy, marked as *ours* in Tab. 1, while at the same time delivering the second-highest completeness, behind [7] which has much lower accuracy. For this setting we employ $f_\epsilon = 0.1$ and $f_{con} = 3$ for fusion.

There is always a trade-off between accuracy and completeness, which depends on how strict one sets the thresholds for rejecting uncertain matches. We thus also run

		Accuracy		Completeness	
		Mean	Med.	Mean	Med.
Points	ours	**0.273**	**0.196**	0.687	0.260
	ours comp	0.379	0.234	**0.400**	0.188
	ours fast	0.289	0.207	0.841	0.285
	tola [36]	0.307	0.198	1.097	0.456
	furu [12]	0.605	0.321	0.842	0.431
	camp [7]	0.753	0.480	0.540	**0.179**
Surfaces	ours	**0.363**	**0.215**	0.766	0.329
	ours comp	0.631	0.262	**0.519**	**0.309**
	ours fast	0.358	0.221	0.939	0.350
	tola [36]	0.488	0.244	0.974	0.382
	furu [12]	1.299	0.534	0.702	0.405
	camp [7]	1.411	0.579	0.562	0.322

Table 1: Quantitative comparison with three different settings on the DTU dataset [22]. The quality metrics accuracy and completeness were computed in accordance to [22], stating the mean and median error in millimeters.

our method with different setting for the fusion, chosen to achieve high completeness (*ours comp* in Tab. 1) with $f_\epsilon = 0.3$ and $f_{con} = 2$. In that setting we surpass [7] in terms of completeness, while still achieving the second best accuracy, slightly below [36] which is a lot sparser. The runtime is ≈**50 seconds** per depthmap when the number of selected views is 10, growing linearly as more views are considered.

Speed settings To explore the behavior of our method when tuned for high speed, we also tried an extreme setting. To speed up the reconstruction we set the window size to 15, restrict the similarity computation in the window to every fourth row and column, and use at most 10 (randomly chosen) views within the view selection criteria for a depthmap. Furthermore, we stop the propagation after 6 iterations instead of 8 and use a reduced set of update candidates in the propagation step, as depicted in Fig. 2c. For fusion we used the parameters $f_\epsilon = 0.3$ and $f_{con} = 3$. With these settings, the method needs ≈**2.7 seconds** per depthmap on the GPU, respectively 7 minutes per complete object (including disk I/O). These extreme settings do lead to some loss in completeness, whereas the accuracy remains high. Even when tuned for speed the method is competitive with the state of the art, see row *ours fast* in Tab. 1.

Fig. 6 presents a qualitative comparison of the three different parameter settings presented.

5.2. Middlebury

We evaluated our method also on the popular Middlebury multiview benchmark [34]. It was the first benchmark for multiview reconstruction, and by now is rather saturated. Moreover, the images are rather small at 640×480 pixels,

	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
Sort By	[mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]
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
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
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
Schroers	0.57	99.1	0.64	96.4	2.12	62.9	0.33	99.7	0.33	99.7	0.54	98.6
Guillemaut	0.43	99.0	0.71	97.6	0.86	96.2	0.35	100	0.58	99.5	0.68	98.0
Sort By	[mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]	[mm]	[%]
BMVC2014_183	0.34	99.4					0.42	98.1				
Hernandez	0.36	99.7	0.52	99.5	0.75	95.3	0.49	99.6	0.45	97.9	0.6	98.5
Mücke	0.36	99.7	0.46	99.1								
Fuhrmann-SG14	0.39	99.4										
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

Figure 7: Screenshots from the Middlebury evaluation for *Dino Full* and *Temple Full*, sorted by accuracy at the standard threshold of 90%. Our method is highlighted in yellow.

Figure 8: Ground truth surfaces and reconstructions for *Temple Full* and *Dino Full* of the Middlebury multiview stereo evaluation dataset [34].

and the dataset consists of only two objects, in three different settings with varying number of views. Fig. 8 shows our reconstructions of the two objects *Dino* and *Temple* ("full" setting with more than 300 views each).

On *Dino* we rank 1st for the "full" version, 3rd for the "ring" version, and 7th for the "sparse" version. On *Temple* we rank 5th on "full", 7th on "ring" and 5th on "sparse". For all six reconstructions the completeness lies between 97.0 and 99.9%, see Fig. 7. It is interesting to note that several methods perform well only on one of the two objects. We achieve excellent performance on both datasets, in terms of both accuracy and completeness. To generate a depthmap from 10 views for (resolution of 640×480) our method needs 2.5 seconds, with window size 11.

5.3. Outdoor Images

We have also tested our method on oblique aerial images from the city of Enschede. Oblique viewing angles are becoming popular in aerial mapping to cover vertical structures such as building facades, and are an ideal testbed for us. Aerial mapping images are routinely recorded in such a way that multiple overlapping views are available. They present a challenge for conventional matching methods, because the depth range is much larger compared to conventional nadir views. And they deviate strongly from

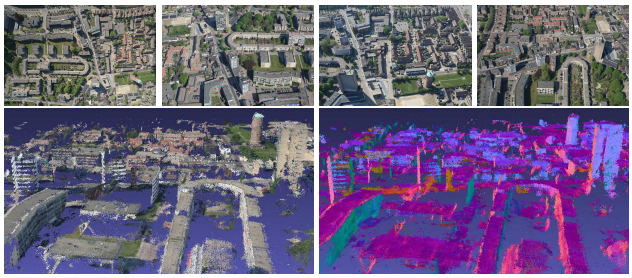


Figure 9: Point clouds generated from aerial images. *Top:* selection of input images. *Bottom:* textured and normal color coded point clouds.

the fronto-parallel setup assumed by many matching algorithms, but do tend to have piecewise constant normals, *e.g.* on the ground, building walls, roofs *etc*. The results in Fig. 9 highlight the properties of *Gipuma*: planes in general orientation are recovered without stair-casing artifacts (see the reconstruction with color coded normals); matching is less reliable on structures without well-defined normals (*e.g.* trees in the near-field) but errors are detected and removed.

We show additional results for the stereo data of the KITTI Vision Benchmark Suite [15] in the supplementary material.

6. Conclusion

We have presented ***Gipuma***, a massively parallel multiview extension of Patchmatch stereo. The method features a new red-black propagation scheme tailored to modern GPU processing, and exploits multiview information during matching. Switching to a common 3D scene coordinate system, in which all camera poses reside, makes it possible to directly integrate information from multiple views in the matching procedure, via the planar homographies induced by a point and its associated normal vector. Like the original Patchmatch stereo, the method is based on slanted planes, and therefore allows slanted support windows without fronto-parallel bias. It is thus particularly well-suited for the frequent case of locally smooth scenes with large depth range. As a by-product, the resulting multiview matcher delivers not only dense depth maps, but also dense normal vectors in metric scene space. Our method achieves accurate and complete reconstruction with low runtime. Quantitative results on the DTU and Middlebury benchmark confirm that it is both more accurate and much faster than state-of-the-art methods such as PMVS. ***Gipuma*** is released to the community as open-source software[1].

[1] www.igp.ethz.ch/photogrammetry/research/gipuma

References

[1] C. Bailer, M. Finckh, and H. P. Lensch. Scale robust multi view stereo. *ECCV 2012*.

[2] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving patchmatch for large displacement optical flow. *CVPR 2014*.

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *SIGGRAPH 2009*.

[4] F. Besse, C. Rother, A. Fitzgibbon, and J. Kautz. PMBP: PatchMatch belief propagation for correspondence field estimation. *BMVC 2012*.

[5] M. Bleyer, C. Rhemann, and C. Rother. PatchMatch Stereo - stereo matching with slanted support windows. *BMVC 2011*.

[6] P. Burt, L. Wixson, and G. Salgian. Electronically directed "focal" stereo. *ICCV 1995*.

[7] N. D. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. *ECCV 2008*.

[8] R. T. Collins. A space-sweep approach to true multi-image matching. *CVPR 1996*.

[9] F. Devernay and O. Faugeras. Computing differential properties of 3-d shapes from stereoscopic images without 3-d models. *CVPR 1994*.

[10] N. Einecke and J. Eggert. Stereo image warping for improved depth estimation of road surfaces. *Intelligent Vehicle Symposium 2013*.

[11] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *IJCV*, 70(1), 2006.

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

[13] A. Fusiello, V. Roberto, and E. Trucco. Efficient stereo with multiple windowing. *CVPR 1997*.

[14] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. *CVPR 2007*.

[15] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. *CVPR 2012*.

[16] R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, second edition, 2004.

[17] P. Heise, S. Klose, B. Jensen, and A. Knoll. PM-Huber: PatchMatch with Huber regularization for stereo matching. *ICCV 2013*.

[18] H. Hirschmüller. Stereo processing by semi-global matching and mutual information. *IEEE TPAMI*, 30(2):328–341, 2008.

[19] X. Hu and P. Mordohai. Least commitment, viewpoint-based, multi-view stereo. *3DIMPVT 2012*.

[20] S. S. Intille and A. F. Bobick. Disparity-space images and large occlusion stereo. *ECCV 1994*.

[21] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. *CVPR 2011*.

[22] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and A. s H. Large scale multi-view stereopsis evaluation. *CVPR 2014*.

[23] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE TPAMI*, 16(9):920–932, 1994.

[24] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. *CVPR 2001*.

[25] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. *Eurographics Symposium on Geometry Processing 2006*.

[26] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):29, 2013.

[27] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. *ICCV 2001*.

[28] P. Labatut, J.-P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. *ICCV 2007*.

[29] G. Marsaglia. Choosing a point from the surface of a sphere. *Annals of Mathematical Statistics*, 43(2):645–646, 1972.

[30] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE TPAMI*, 15(4):353–363, 1993.

[31] R. Ranftl, S. Gehrig, T. Pock, and H. Bischof. Pushing the limits of stereo using variational stereo estimation. *Intelligent Vehicles Symposium 2012*.

[32] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *CVPR 2011*.

[33] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, 2002.

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

[35] S. Shen. Accurate multiple view 3d reconstruction using patch-based stereo for large-scale scenes. *IEEE TIP*, 22(5):1901–1914, 2013.

[36] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE TPAMI*, 32(5):815–830, 2010.

[37] E. Tola, C. Strecha, and P. Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *MVA*, 23(5):903–920, 2012.

[38] K.-J. Yoon and I. S. Kweon. Adaptive support-weight approach for correspondence search. *IEEE TPAMI*, 28(4):650–656, 2006.

[39] C. Zach. Fast and high quality fusion of depth maps. *3DPVT 2008*.

[40] E. Zheng, E. Dunn, V. Jojic, and J.-M. Frahm. PatchMatch based joint view selection and depthmap estimation. *CVPR 2014*.

[41] C. Zinner, M. Humenberger, K. Ambrosch, and W. Kubinger. An optimized software-based implementation of a census-based stereo matching algorithm. In *Advances in Visual Computing*, pages 216–227. 2008.