Multi-View Stereo Point Clouds Visualization

Yi Gong and Yuan-Fang Wang

Computer Science Department, University of California, Santa Barbara, CA 93106

Abstract. 3D reconstruction from image sequences using multi-view stereo (MVS) algorithms is an important research area in computer vision and has multitude of applications. Due to its image-feature-based analysis, 3D point clouds derived from such algorithms are irregularly distributed and can be sparse at plain surface areas. Noise and outliers also degrade the resulting 3D clouds. Recovering an accurate surface description from such cloud data thus requires sophisticated post processing which can be computationally expensive even for small datasets. For time critical applications, plausible visualization is preferable. We present a fast and robust method for multi-view point splatting to visualize MVS point clouds. Elliptical surfels of adaptive sizes are used for better approximating the object surface, and view-independent textures are assigned to each surfel according to MRF-based energy optimization. The experiments show that our method can create surfel models with textures from low-quality MVS data within seconds. Rendering results are plausible with a small time cost due to our view-independent texture mapping strategy.

1 Introduction

Research in 3D model reconstruction using multi-view stereo (MVS) algorithms has made significant strides recently in the computer vision community. As a result, 3D point cloud data are no longer derivable only from expensive and specialized devices like range scanners, but also from uncalibrated consumer-market digital cameras [1] or even community photo collections [2]. However, 3D point clouds recovered by these computer vision algorithms are not as ideal as those from specialized scanners. MVS algorithms use feature-based analysis under the Lambertian shading assumption, which have difficulty handling textureless and non-Lambertian surfaces. Combined with the inherent difficulties of solving an ill-posed, inverse 2D to 3D problem, 3D point clouds reconstructed from MVS algorithms are often sparse, irregularly distributed, noisy, and usually contain many outliers (see Figure 1). Such datasets present challenges to the recovery of object surface structure and appearance.

To improve the quality of the recovered 3D point clouds, post-processing steps are often employed to adjust the positions and orientations of 3D points based on photo discrepancy constraints [1]. But such methods can be computationally very expensive. For applications requiring fast 3D model reconstruction, e.g., creating avatars for game players, a long waiting time is unacceptable. Efficient

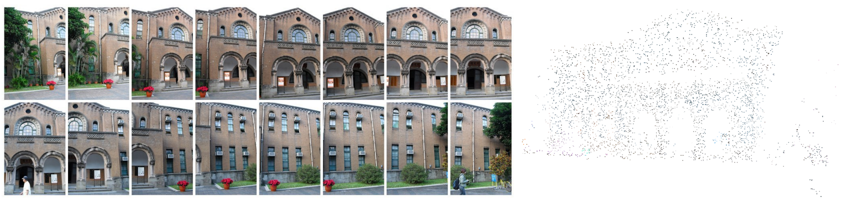

Fig. 1. An example of MVS data. Left: input images; right: output point clouds.

and robust visualization methods for such MVS 3D datasets are important in completing the pipeline of image-based 3D reconstruction. Although computer graphics researchers have explored point-based modeling and rendering techniques for a long time, most existing work is not applicable to the raw 3D point data derived from MVS algorithms due to the data quality issues. Furthermore, deriving consistent texture maps for MVS data presents additional research issues of avoiding color mismatch and cracking using the color/texture information from multiple images.

This paper is aimed at providing an efficient and generic solution for visualizing unpolished MVS point clouds. Our main contributions are: we propose a statistical method to select nearest neighbors in highly irregularly-distributed point clouds; we apply adaptive radius elliptically-shaped surfels to approximate object surfaces; we introduce Markov Random Field (MRF) based multi-view texture mapping for point splatting. Our results show the proposed methods can handle such challenging 3D datasets much better than existing ones.

2 Related Work

For relatively clean, regular and dense point clouds, many existing algorithms have been developed to extract the geometric surface precisely. They can be divided into two categories: *explicit* and *implicit* methods. Explicit methods make use of Voronoi diagram and Delaunay triangulation to connect points and form surface meshes [3,4,5]. As explicit methods require dense point clouds as inputs and are sensitive to noise and outliers, they are not suitable to process irregularly distributed 3D point data generated from the MVS algorithms. Implicit methods do not connect points explicitly. Instead, they define an inside/outside function $f()$ and take $f = 0$ as the surface. This $f()$ function is determined by minimizing an energy expression related to the distances from the input points P to the surface function $f()$. Hoppe et al. first used the distance field to extract surfaces from point clouds [6]. Later, radial basis functions [7,8,9], moving least squares [10], level set [11] and poisson functions [12] were also introduced to solve the implicit function. Implicit methods are relatively robust to noisy data, but require an additional triangulation step to transform implicit functions to polygon meshes. Moreover, they are sensitive to normal estimation errors, which is a common problem for MVS data due to their sparsity and irregularity. Finally, for complex scenes containing overlapping objects, implicit surface extraction

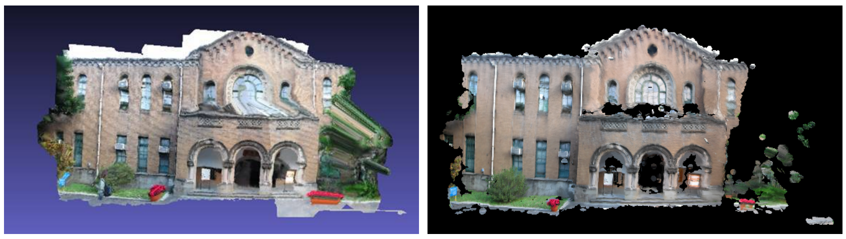

Fig. 2. Left: incorrect surface connection generated by implicit surface extraction; right: the result of our method

techniques will generate unexpected inter-object connections between unrelated points, as shown in Figure 2(left).

To simplify the visualization of 3D point cloud data, Pfister et al. [13] and Zwicker et al. [14] proposed the EWA point splatting method which skips the surface extraction step and visualizes an object's surface directly with oriented disks, called *surfels* – surface elements. Surfels overlap with each other in 3D space to cover gaps among the points. When surfels are projected onto the screen, pixels covered by multiple surfels will blend their surface attributes such as normals and textures in a weighted manner. This no-topology method avoids incorrect connections of points and thus is very suitable for MVS data visualization, whose points' group division is especially difficult when close or overlapping objects present. Goesele et al. proposed an ambient point cloud concept that visualizes difficult backgrounds with ambient points to reduce artifacts in view interpolation [15]. But their method is more focused on the view interpolation effect and is different from our purpose.

Before MVS algorithms had got mature enough to enable automatic image-based reconstruction, quite a few researchers tried image-based rendering (IBR) methods to visualize 3D scene based on multi-view images while totally discarding assumptions of the scene geometries [16]. But these techniques usually require a huge amount of images and have blurred results due to their blending essence for new viewpoints. Debevec et al. [17] introduced view-dependent texture mapping (VDTM) with manually built 3D geometric model with much less input images to solve these problems. VDTM inherits the ability of replaying view-dependent effects from pure IBR methods. But due to its visibility based subdivision step, the model becomes so complex that requires huge memory to store, and consumes more time to render than view-independent methods. Yang et al. combined VDTM with point splatting [18]. But their experiments were mainly tested on synthetic/calibrated systems which are more clean and precise than our input, and is suitable to apply VDTM algorithm directly. For MVS reconstructed 3D points from uncalibrated camera like ours, blending multi-view textures cannot eliminate the appearance incoherence among neighboring points and neighboring views due to re-projection error and exposure difference, which are the most serious problems of MVS data.

Lempitsky et al. [19] presented a view-independent texture mapping approach by minimizing an energy expression that incorporates both viewing direction matching and texture coherence. The selected image to texture a mesh triangle should have its viewing direction close to the triangle's normal. Meanwhile, neighboring triangles are expected to sample textures from the same image. They use MRF optimization to constrain the relationship between these two factors. We also use MRF optimization to select texture based on the energy of surface orientation and coherence among neighbors. But point clouds don't have explicit connections like polygon meshes to build the MRF directly. Our strategy is to create a graph conservatively, based on the distances between points and their sizes. The details will be explained in section 3.3.

3 Our Method

3.1 Nearest Neighbors

Picking nearest neighbors is the first and key step for surface approximation from point clouds, because the surfels' orientations, sizes and positions are mainly determined by the information from their neighbors. As we mentioned above, for MVS data, the 3D points are actually recovered from 2D features detected in input images. The distribution of the these image features is usually very sparse and uneven at featureless regions. Traditionally, each point is assigned a fixed radius for neighbor selection, but such a strategy is not ideal when applied to MVS data, because it may select either too few or too many neighbors. Fixing the number of nearest neighbors is not a good design either, as it may include some neighbors very far away that contribute little or even incorrect information to the current point's local geometry.

To collect an adequate and representative set of neighbors for normal estimation, we allow the number of nearest neighbors to vary in a range and make the decision of accepting or rejecting a neighbor by maximizing the relative difference between the accepted group of neighbors and the discarded ones. Suppose we have n nearest neighbors sorted by distance in ascending order. We will accept the first

$$k^* = \arg\max_k \frac{d_{k+1} - \hat{\mu}_k}{\hat{\sigma}_k / \sqrt{k}} \tag{1}$$

neighbors and reject the range from $k^* + 1$ to n. In the equation, d_{k+1} is the distance of the $(k+1)$th neighbor, and $\hat{\mu}_k$ and $\hat{\sigma}_k$ are the sample mean and sample standard deviation of the distances of the first k nearest neighbors. The formula above to maximize is essentially reduced from Welch's t-test statistic [20]

$$t_{A,B} = \frac{\hat{\mu}_A - \hat{\mu}_B}{\sqrt{\hat{\sigma}_A^2/n_A + \hat{\sigma}_B^2/n_B}} \tag{2}$$

which serves as a discriminant measure between the two sample groups $A = \{d_1, d_2, \cdots, d_k\}$ and $B = \{d_{k+1}\}$ for our case, where the sample sizes $n_A = k$ and $n_B = 1$, and the second sample has only one element, thus a degenerate

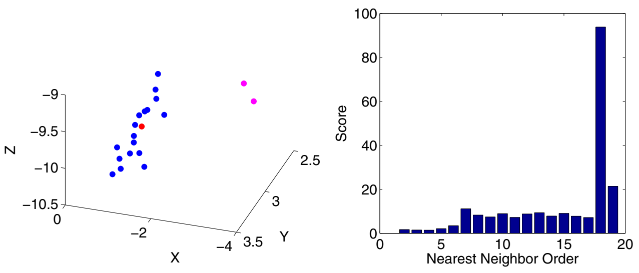

Fig. 3. An example of our neighbor selection. Left: a 3D point (in red) and its accepted (in blue) and rejected (in pink) neighbors; Right: scores of the tested neighbors.

mean $\hat{\mu}_B = d_{k+1}$ and a zero variance $\hat{\sigma}_B^2 = 0$. The effect of this algorithm is shown in Figure 3. The current point is marked in red. The remaining dots are its nearest neighbors sorted by distance from it. From the score chart, obviously the 18th neighbor get the highest score, i.e. k^* equals 18. The rest two points are rejected, which are marked in pink.

3.2 Adaptive-Size Elliptical Surfels

After obtaining an estimate of a local neighborhood above, we approximate the object surface by oriented surfels. The normal of each surfel is available through Principal Component Analysis (PCA) [6], which is relatively robust for MVS data. But the size and shape need more sophisticated approach to better match local neighborhood geometry. Due to the irregular point distribution, each surfel's size need to be adaptive to the local cloud density to cover the gap among the points. Most existing point-based rendering algorithms use circular disks as the rendering primitives, which does not work well with MVS data. For example, for points on the ridge of a sharp geometry (i.e., an edge), large surfels will jut out from approximated surface and look abrupt and wired. Thus, the surfel sizes should also relate to the local curvatures in different directions . For flat surface patches that have a small absolute value of surface curvature, a surfel can extend its area in any direction until it covers the gap in that direction or reaches the maximum radius we set. For highly curved patch, surfel radius should be small. For patches flat in some directions and highly curved in other directions, we should assign anisotropic radii to the surfels.

From differential geometry, we know that the largest difference between two surface curvatures of a point occurs at two perpendicular directions on the tangent plane, i.e. the principal directions [21]. Therefore, an elliptical surfel with two different axes which can be adjusted independently perfectly meets our needs. To estimate the local curvature, we fit a least square quadratic surface over the given point and its nearest neighbors. Then the principal curvatures can be derived from Gaussian curvature and mean curvature [21] calculated based on the surface equation. As for the principal directions, we adopt Che et al.'s method [22]. The radii along the two axes are assigned to $\min\left(\frac{1}{\kappa}, \frac{1}{k}\sum_{i=1}^{k} d_i\right)$, where κ is the curvature of the current point, whose reciprocal equals to the radius of the tangent circle in the given direction at that point. The second term is an estimation of the sparsity of the current point's neighborhood, which is the average distance of the first k neighbors to current point. In our implementation, we have used $k = 5$ based on experiments. We also set a minimum and maximum limitation for the surfel radius to avoid extreme cases. With our algorithm, we can fill gaps among points better and avoid improperly extended surfels.

3.3 Multi-View Texture Mapping

Texture coherence among neighboring points is very important to MVS data visualization because of the non-ignorable re-projection error of reconstructed 3D points and exposure difference among different view images. Seam and mismatch are inevitable if neighboring surface points take different images as textures. As a result, simply choosing the closest orientation image as the texture source for each surfel will not lead to a satisfactory result (Figure 4). Therefore, we need to strike a balance between similarity in viewing direction and coherence of neighborhood, i.e., the texture source, among neighboring surfels. Since this coherence is only related to neighboring surfels, which conforms to Markovian property, we can define this problem as an energy minimization problem over an MRF. In contrast to Lempitsky and Ivanov's method [19], which applies MRF to optimize texture mapping of polygon meshes, we do not have an explicit neighbor graph. Our points's neighbor relationship is vague and implicit because our primitives have no explicit connections and the surfel's radius is adaptive. We judge two points p_1 and p_2 to be neighbors if their distance is smaller than the sum of their long axes. This is a conservative estimation. But as we want the color to transfer smoothly, making close points share similar texture sources is not a drawback. According to this rule, we add edges between surfel pairs to create the MRF. Its energy function has two parts: data energy which only relates to the given point's visibility in a certain image (already known in typical MVS result data) from MVS data and how close the camera's viewing angle is to the point's normal; and smoothness energy which penalizes if current surfel does not use the same texture as its neighbors.

$$E(p) = \mathrm{data}(p, \mathrm{camera}_i) + \lambda\, \mathrm{smoothness}(p, \mathrm{neighbor\_of}(p)) \tag{3}$$

in which λ is a regularization parameter to balance between these two terms. We use graph-cuts [23, 24, 25] to solve the MRF optimization and it works well.

3.4 Point Splatting

Our rendering method is similar to Botsch et al.'s [26]. We first disable the color buffer's writing operation and enable the depth buffer's to render surfels into the depth buffer without shading. Then the depth buffer is disabled and the color buffer is enabled. We shift all surfels a little closer to the camera to

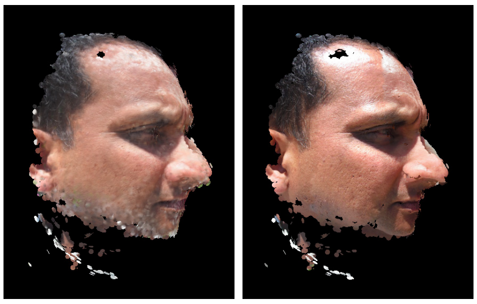

Fig. 4. Multi-view texture mapping without (left) and with (right) MRF optimization

avoid z-fighting and render them with the assigned textures onto the screen. The colors and weights of covered pixels are accumulated by alpha blending in the frame buffer object, where weights are recorded in the alpha channel. Finally, each pixel's color is normalized by its alpha value. This splatting process also helps us smooth the boundaries between surfels, as overlapping surfels will all contribute to the intersection area and blur the boundary naturally.

4 Experiments

We test our algorithms on a computer configured with a Core 2 Quad 2.33GHz CPU, 4GB DDR3 memory, and an Nvidia GeForce 210 video card with 512M video memory. Our input is the 3D point clouds generated by the SfM algorithm [27]. Note that the algorithm uses photos from uncalibrated, consumer-market digital cameras with arbitrary photographing modes. Illumination and exposure are not controlled in these sequences and adjacent images can exhibit large change in color and brightness. The camera's intrinsic and extrinsic parameters are recovered by a non-linear optimization using corresponding feature points in different images. Hence, the point clouds have errors and outliers. The data contains sparse but precisely recovered 3D points from SIFT features and dense but flawed points recovered by interpolation. When calculating normals and curvatures, we give high weight to the SIFT features and low weight for interpolated points to reduce error.

Though we only test performance on the data sets from this MVS algorithm, our method is applicable to point clouds from other MVS algorithms too. 3D points generated from MVS algorithms have similar attributes and formats: 3D positions, colors, 2D coordinates in visible images. Camera parameters are also available from their own recovering process. So all the information we need are accessible in typical MVS algorithms.

In our experiments, the operations of searching nearest neighbors [28], calculating normal, and estimating principal curvatures are very fast. Optimizing texture selection with MRF is the most time consuming stage in our algorithms. We run 2 iterations of alpha expansion with graph-cuts which cost 14 to 18 seconds for 41k to 50k point models in our tests (Table 1). From our results, we

Table 1. Our test datasets and corresponding time costs

Model	# PTS	# Views	# MRF Edges	# Used Views	Time(fitting)	Time(texture)
Building	60,206	16	1,039,649	15	3.132s	15.253s
Logcabin	41,303	29	683,124	18	2.142s	13.844s
Nandhu	43,247	22	712,329	14	2.301s	17.697s
Tobias	49,044	21	809,571	14	2.366s	20.614s
Medusa	63,373	19	1,026,093	15	2.970s	33.159s

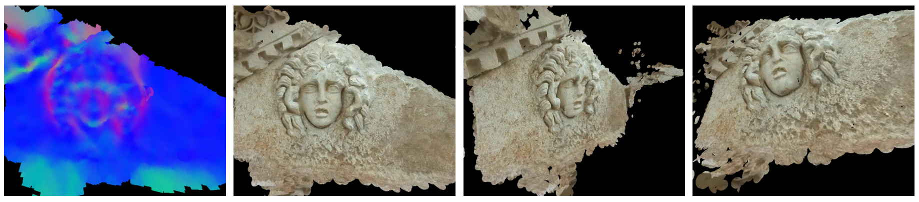
(a) (b) (c) (d)

Fig. 5. Medusa model. (a) recovered normals; (b)(c)(d) results from different views.

can see that even without very precisely estimated normals, MRF based texture mapping strategy can still produce plausible visualization results, while selecting texture using only the best matched viewing direction leads to color jump and discontinuity (Figure 4).

We also compare our result with Autodesk's photofly [29], the state-of-the-art application of 3D reconstruction and visualization. Photofly has its own 3D reconstruction algorithm. So its input point clouds may have slight difference from ours. But it is still a good reference for the visualization part. From Figures 6 and 7, we can see the large difference between its rendering strategy and ours. Photofly blends multi-view textures to reduce the artifacts brought by low quality MVS data, which blurs the picture heavily, while we render surfels boldly without vagueness after optimizing their orientation, shape and textures.

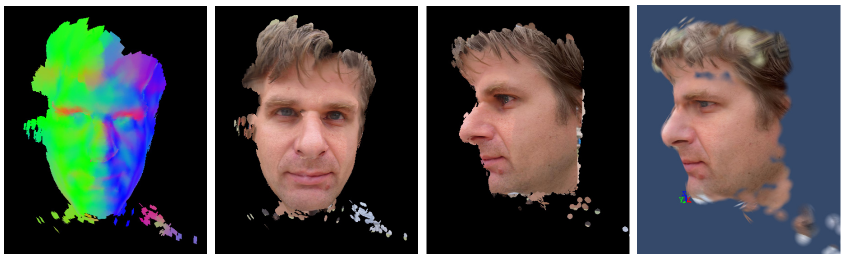
(a) (b) (c) (d)

Fig. 6. Tobias model. (a) recovered normals; (b)(c) results from different views; (d) Photofly's [29] rendering result - in the hair area, the artifacts caused by multi-view images blending is very obvious.

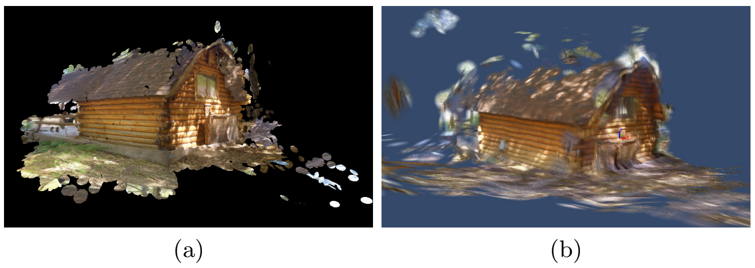

Fig. 7. Logcabin model. Rendering results of (a) Ours and (b) Photofly's [29].

5 Conclusions

We have presented a novel algorithm to visualize challenging point clouds generated from the MVS algorithms. Our algorithm uses a statistical metric to select representative neighbors for each 3D point and then approximates the point's neighborhood with an elliptical surfel of an adaptive size. Each surfel's orientation, shape and size are calculated according to its neighborhood information. To remedy the imprecision of an approximated surface, we apply MRF optimized texture mapping strategy to select most proper images as texture source for each surfel while minimizing appearance incoherence among neighboring surfels. Our results show the proposed algorithm handles the low quality 3D data well. Our future work includes improving the precision of the normal estimation by taking image information into consideration and exploring possible shape detection on MVS data to make the surface reconstruction more robust.

References

1. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. In: Proc. CVPR 2007, pp. 1–8 (2007)
2. Goesele, M., Snavely, N., Curless, B., Hoppe, H., Seitz, S.M.: Multi-view stereo for community photo collections. In: Proc. ICCV 2007, pp. 1–8 (2007)
3. Edelsbrunner, H., Mücke, E.P.: Three-dimensional alpha shapes. In: Proc. VVS 1992, pp. 75–82 (1992)
4. Amenta, N., Bern, M., Kamvysselis, M.: A new voronoi-based surface reconstruction algorithm. In: Proc. SIGGRAPH 1998, pp. 415–421 (1998)
5. Amenta, N., Choi, S., Kolluri, R.K.: The power crust. In: Proc. SMA 2001, pp. 249–266 (2001)
6. Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., Stuetzle, W.: Piecewise smooth surface reconstruction. In: Proc. SIGGRAPH 1994, pp. 295–302 (1994)
7. Turk, G., O'Brien, J.F.: Shape transformation using variational implicit functions. In: Proc. SIGGRAPH 1999, pp. 335–342 (1999)
8. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3d objects with radial basis functions. In: Proc. SIGGRAPH 2001, pp. 67–76 (2001)

9. Ohtake, Y., Belyaev, A., Seidel, H.P.: Ridge-valley lines on meshes via implicit surface fitting. In: Proc. SIGGRAPH 2004, pp. 609–612 (2004)
10. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Computing and rendering point set surfaces. IEEE Trans. Visual. Comput. Graph. 9, 3–15 (2003)
11. Zhao, H.K., Osher, S., Fedkiw, R.: Fast surface reconstruction using the level set method. In: Proc. VLSM 2001, pp. 194–201 (2001)
12. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proc. SGP 2006, pp. 61–70 (2006)
13. Pfister, H., Zwicker, M., van Baar, J., Gross, M.: Surfels: surface elements as rendering primitives. In: Proc. SIGGRAPH 2000, pp. 335–342 (2000)
14. Zwicker, M., Pfister, H., van Baar, J., Gross, M.: Surface splatting. In: Proc. SIGGRAPH 2001, pp. 371–378 (2001)
15. Goesele, M., Ackermann, J., Fuhrmann, S., Haubold, C., Klowsky, R., Steedly, D., Szeliski, R.: Ambient point clouds for view interpolation. In: Proc. SIGGRAPH 2010, pp. 95:1–95:6 (2010)
16. Shum, H.Y., Chan, S.C., Kang, S.B.: Image-Based Rendering. Springer, Heidelberg (2006)
17. Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In: Proc. SIGGRAPH 1996, pp. 11–20 (1996)
18. Yang, R., Guinnip, D., Wang, L.: View-dependent textured splatting. The Visual Computer 22, 456–467 (2006)
19. Lempitsky, V., Ivanov, D.: Seamless mosaicing of image-based texture maps. In: Proc. CVPR 2007, pp. 1–6 (2007)
20. Welch, B.L.: The generalization of "student's" problem when several different population variances are involved. Biometrika 34, 28–35 (1947)
21. Struik, D.J.: Lectures on Classical Differential Geometry. Addison-Wesley, Reading (1950)
22. Che, W., Paul, J.C., Zhang, X.: Lines of curvature and umbilical points for implicit surfaces. Computer Aided Geometric Design 24, 395–409 (2007)
23. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. 23, 1222–1239 (2001)
24. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell. 26, 1124–1137 (2004)
25. Kolmogorov, V., Zabin, R.: What energy functions can be minimized via graph cuts? IEEE Trans. Pattern Anal. Mach. Intell. 26, 147–159 (2004)
26. Botsch, M., Hornung, A., Zwicker, M., Kobbelt, L.: High-quality surface splatting on today's gpus. In: Proc. PBG 2005, pp. 17–141 (2005)
27. Chen, C.I., Sargent, D., Tsai, C.M., Wang, Y.F., Koppel, D.: Stabilizing stereo correspondence computation using delaunay triangulation and planar homography. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y.K., Rhyne, T.-M., Monroe, L. (eds.) ISVC 2008, Part I. LNCS, vol. 5358, pp. 836–845. Springer, Heidelberg (2008)
28. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.: An optimal algorithm for approximate nearest neighbor searching. J. ACM 45, 891–923 (1998)
29. Photofly, A., `http://labs.autodesk.com/technologies/photofly/`