Photo-consistent Planar Patches from Unstructured Cloud of Points

Roberto Toldo and Andrea Fusiello

Dipartimento di Informatica
Università di Verona
Strada Le Grazie 15, 37134 Verona (Italy)
{roberto.toldo,andrea.fusiello}@univr.it

Abstract. Planar patches are a very compact and stable intermediate representation of 3D scenes, as they are a good starting point for a complete automatic reconstruction of surfaces. This paper presents a novel method for extracting planar patches from an unstructured cloud of points that is produced by a typical structure and motion pipeline. The method integrates several constraints inside J-linkage, a robust algorithm for multiple models fitting. It makes use of information coming both from the 3D structure and the images. Several results show the effectiveness of the proposed approach.

1 Introduction

While the current state of the art in architectural three-dimensional (3D) reconstruction has focused on the recovery of dense and accurate representations of objects imaged through pictures or video, the sustained interest in accessible architectural modeling software is a strong evidence of an untapped general need for compact, abstract representations of architectural objects. What separates unstructured cloud of points from higher-level renditions of an architectural model is a *semantic gap*, which should be bridged exploiting additional information. This is one of the most challenging research area in Computer Vision. The proposed methods can be divided in three main categories: interactive, top-down and bottom-up.

Interactive approaches require user intervention to recognize higher level structures, usually basing on the three-dimensional information previously extracted [1–4]. Top-down or model based approaches start from the prior knowledge of the set of potential parametric models and try to infer the best fitting one along with its parameters [5–9]. Potentially, only one image could be employed if the prior knowledge is enough to derive the 3D model [10, 11]. When no prior knowledge is assumed or user intervention is not available, bottom-up methods are employed. They start directly from raw three-dimensional data points trying to aggregate them in progressively higher level structures, possibly using also the information coming from the images. This paper falls in this category: The aim is to leverage models from unorganized point clouds to an intermediate representation, i.e. planar patches, that narrows the gap between acquisition and manipulation of architectural models.

Some methods try to optimize an initial triangulation using visibility [12] or photo-consistency [13, 14] only, i.e., the fact that a patch corresponding to a solid opaque surface has the same appearance in all the images (modulo some geometric and photometric distortions). They work only with very simple convex polyhedra objects, and they assume all points visible by at least one view. A similar approach was proposed in [15]. A sequential MSAC [16] is employed in order to detect the planes. Image-consistent triangulation is then used within a simulated annealing algorithm to create an optimal surface mesh. In [17] an automatic approach to segment a cloud of points into planes is proposed: It generates plane hypotheses by random sampling the 3D points (inspired by RANSAC) and scores them using photo-consistency. The reported experiments involve extremely simple objects. More recently Moser et al. [18] presented a paper that is able to perform out-of-core simplification of an high quality digital surface model of a city using RANSAC. The density and good quality of the input data are crucial here.

Very recent works proposed to run a Multiview Stereo on the output a SaM pipeline[22, 23]. While the results are visually compelling, they do not point at the problem of the semantic gap, since the output is a dense and less compact representation of the scene.

Besides [12–14] which are very simple, all the above papers share a common part since they extract the planes underlying the scenes using RANSAC (or MSAC) with spatial or photo-consistency information. This seems to be a crucial task, but the sequential application of an algorithm designed for single model extraction, is not suitable, and this becomes clear as soon as one steps from clean, structured data to real, noisy unstructured data, as those coming from a structure and motion (SaM) pipeline[19, 20]. Techniques designed to extract *multiple* instances of a model are required in this case, e.g. J-linkage, which has recently been proposed [21] and proved to be very robust. It will be described in details in section 2.

Our strategy reaps the benefits of most of the aforementioned methods: i) it applies to unorganized large cloud of points, ii) employ a multiple model fitting algorithm (J-linkage) and iii) seamlessly integrates both spatial, visibility *and* photo-consistency information inside it.

The output of our algorithm are triangulated planar patches, which are a very compact and stable intermediate representation of 3D scenes, as they are a good starting point for a complete automatic reconstruction of surfaces.

2 Overview of the J-linkage algorithm

In this section the J-linkage algorithm will be briefly overviewed. More details can be found in [21].

The method is based on random sampling, like RANSAC. Each minimal sample set (MSS) defines a tentative model. Imagine to build a $N \times M$ matrix (Fig. 2) where entry (i, j) is 1 if point i is closer to model j than a threshold ε. Each column of that matrix is the characteristic function of the *consensus set* of a model. Each row is the characteristic function of the *preference set* (PS) of a given point, i.e., indicates which models a points has given consensus to. Points belonging to the same structure will have similar PS, in other words, they will cluster in the conceptual space $\{0,1\}^M$.

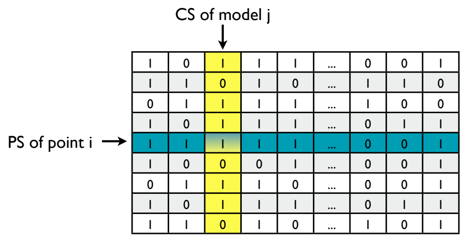


Fig. 1. An example of consensus/preference matrix. Columns are consensus sets (CS), rows are preference sets (PS).

2.1 Random sampling

As in [24] it is assumed that the a-priori probability that two points belong to the same structure is higher the smaller the distance between the points. Hence minimal sample sets are constructed in a way that neighboring points are selected with higher probability. If a point $\mathbf{x}_i$ has already been selected, then $\mathbf{x}_j$ has the following probability of being drawn:

$$P(\mathbf{x}_j|\mathbf{x}_i) = \begin{cases} \frac{1}{Z}\exp -\frac{||\mathbf{x}_j - \mathbf{x}_i||^2}{\sigma^2} & \text{if } \mathbf{x}_j \neq \mathbf{x}_i \\ 0 & \text{if } \mathbf{x}_j = \mathbf{x}_i \end{cases} \tag{1}$$

where Z is a suitable normalization constant and σ is chosen heuristically.

2.2 Agglomerative Clustering

Models are extracted by agglomerative clustering data points in the conceptual space, where each point is represented by its PS. The distance between two elements (point or cluster) is computed as the *Jaccard* distance between the respective preference sets. The PS of a cluster is defined as the intersection of the preference sets of its points. Given two sets A and B, the Jaccard distance is

$$d_{\mathrm{J}}(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}. \tag{2}$$

The Jaccard distance measures the degree of overlap of A and B and ranges from 0 ($A = B$) to 1 ($A \cap B = \emptyset$).

The algorithm proceeds by linking elements with distance smaller than 1 and stops as soon as there are no such elements left. This can be performed efficiently using an heap data structure. As a result, clusters have the following properties:

– for each cluster there exists at least one model that is in the PS of all its points;
– one model cannot be in the PS of *all* the points of two distinct clusters;

The final model parameters for each cluster of points is estimated by least squares fitting.

3 Constraints integration

This paper is aimed at leveraging the J-linkage algorithm to fit planar patches to a cloud of 3D point that are samples of surfaces in the observed scene. Extraction of planar patches is not the same as fitting planes, because a patch is a *region* of the plane, and the same plane may contain more patches (see Fig. 2). The planar patch associated to a set of coplanar points is the convex hull of the projection of the points onto the fitting plane. In order for a planar patch to represent an actual surface, it must satisfy a number of constraints, beside coplanarity, that will be described later. This section will concentrate on how these constraints can be seamlessly integrated inside J-linkage.

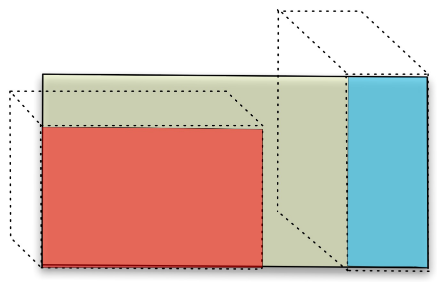


Fig. 2. A single plane (yellow) contains several patches (blue and red).

J-linkage extracts models in an incremental way, by merging smaller structures at each step. In the case on planar patches, two patches can merge only if the result is a set of coplanar points (to some extent). Coplanarity is the invariant property, and any other constraint can be enforced as an invariant property, so that two patches can be merged if and only if the resulting does not violate the constraint.

More in detail, the constraints will be formulated and tested on triangles, since any planar polygon can be triangulated. When two patches are being considered for possible merging, a new patch is computed as the convex hull of the union of the points. By inductive hypothesis the two original patches satisfy the constraints, whereas the new triangles that are created must be tested against the constraints. If a single triangle fails the merging is rejected. A graphical explanation of this incremental step is shown in the Fig. 3.

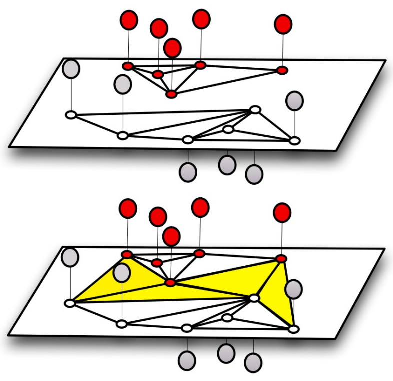


Fig. 3. Incremental step. The constraints are assumed to be valid for each patch (top). When two patches are merged (bottom) the constraints needs to be checked only for the new triangles (yellow).

Three kind of constraints are enforced:

– **Photo-Consistency Constraint**: the projections of a triangle on the images where it is visible should be photo-consistent.
– **Visibility Constraint**: a triangle must not to occlude any visible point.
– **Non Intersection Constraint**: a triangle must not intersect any previously defined surface.

3.1 Photo-Consistency Constraint

A patch in space is *image-consistent* if all its projections onto the images where it is visible contain conjugate points. Image consistent patches are attached to actual object surfaces in the scene (see Fig. 4). Image-consistency can be checked through *photo-consistency*, the property that the projections of a patch are equal up to a projective transformation and photometric nuances.

Let us first define a set of compatible images as the ones where the vertices of a given triangle are visible. Among them, the one where the projected triangle exhibits the maximum area is chosen as the reference. All the triangles in the compatible images are projectively warped onto the triangle in the reference image and compared to it through normalized cross-correlation (NCC). The final photo-consistency of the 3D triangle is obtained as the average of the NCC scores of its projections (the value ranges from -1 to 1), and its is considered photo-consistent if this value is below a fixed threshold.

3.2 Visibility Constraint

A Structure and Motion pipeline generally outputs the *visibility* of the points, i.e. the cameras from which a point is visible. This information can be exploited

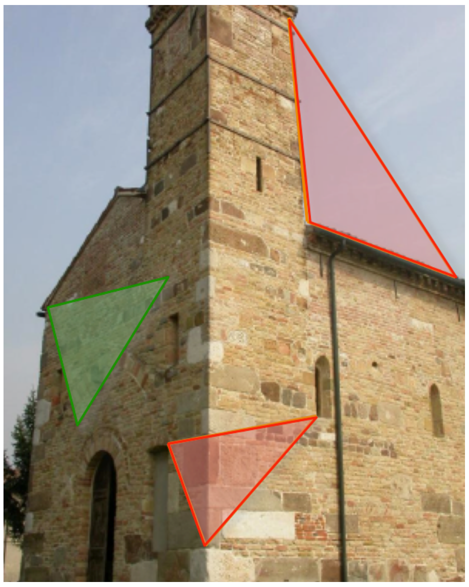


Fig. 4. The green triangle is image-consistent, the red ones are not.

to formulate a simple yet powerful constraint: a surface patch must not occlude a 3D point from the view where it is visible.

Mathematically, this translates into a segment-triangle intersection test. The segment ranges from the optical center of the view to the 3D point that is being examined. The intersection test can be performed efficiently at constant time. However, in the worst case - i.e. when no intersections with the current triangle were found - one need to run the test for each view and for each visible point from that view. In order to speed up the process, we precompute the axis aligned bounding box (AABB) for each view that contains every visible points and the optical center. We also compute and update an AABB that contains every point of a patch. A prior intersection test is made between the AABB of the patch and the AABB of a view: if no intersection occurs we are assured that no triangle of the patch will intersect a segment in that view. The intersection test between two AABB also takes constant time.

3.3 Non Intersection Constraint

During the patch growing, it may happen that patches end up intersecting each other in their interior. This is clearly an unwanted situation, as the customary assumption holds that surfaces are manifolds. To avoid this, we embed the non intersection constraint directly in the J-linkage.

When creating a new patch we check that it is not intersecting any previously defined patch. This translates into a triangle-triangle intersection test among all the triangles of two patches. The triangle-triangle intersection test can be computed in constant time. However, when dealing with surfaces composed by many triangles, it may require many checks. We speed up the process taking advantage of the AABB computed for every patch.

4 Filling the gaps

During the agglomerative clustering of J-linkage, it is sufficient that a single triangle does not satisfy a constraint to discard the entire merge, because it is inductively assumed that patches are *convex*. As a result, triangles that fulfill the constraints are discarded, thereby leaving gaps in the surfaces between neighbouring patches (Fig. 5). This issue is solved *a-posteriori*, by a gap-filing heuristics that relaxes the convexity assumption.

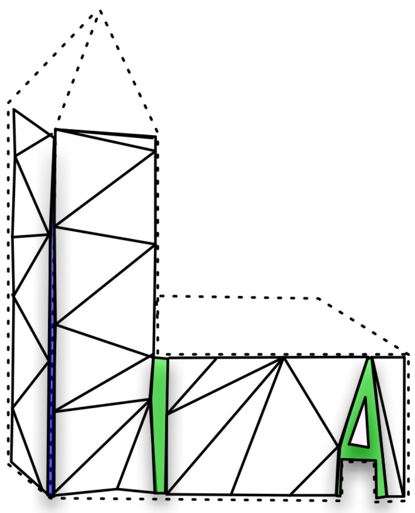


Fig. 5. Green regions are gaps between adjacent patches that are to be filled. Blue regions are gaps between orthogonal patches.

Two patches are said to be *adjacent* if at least one of the points of one patch contains a point of the other patch in his k-neighborhood. We can distinguish two cases of adjacent patches: *coplanar*, when the angle between the respective support planes is less than 30 degrees, and *orthogonal*, when the angle lays between 60 and 120 degrees. A graph of connection between the patches can thus be inferred. First, we fill the gaps between orthogonal patches. By construction, a point can belong to only one patch. We identify the points compatible, by means of the inlier threshold, to both the orthogonal patches. The points are then added to both patches if the constraints defined before are valid for the newly computed patches.

Finally, we fill the gaps between coplanar patches by testing each one connecting triangles between the patches using the same methods and constraints defined before. When two patches have been processed they are treated as a single entity in this iterative procedure.

5 Results

We tested our method on real data coming from a completely automatic SaM pipeline. The first test compares our approach to [13] and [21] on a simple object. The second set of experiments demonstrate how our method can cope with real-world examples.

5.1 Comparative test

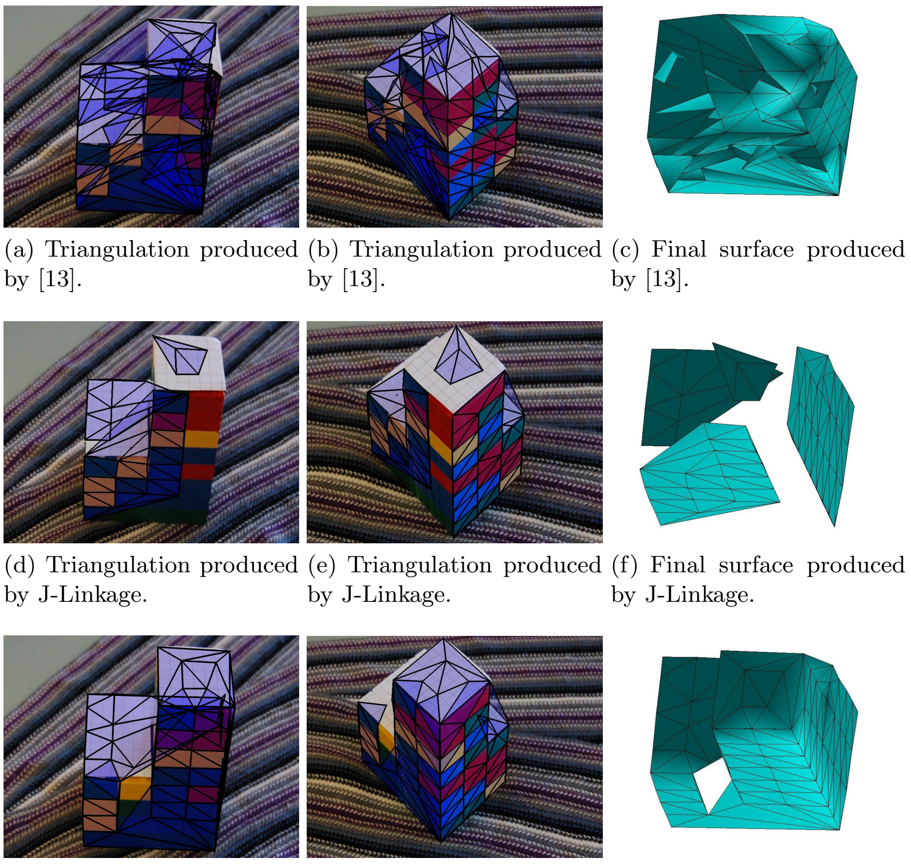
In order to be able to tun a comparative test with [13] we had to choose a setup where all the points projects in all the views. To this end we constructed the "Duplo" object visible in Fig. 6 and manually selected 72 keypoints correspondences in 5 views. The 3D structure have been recovered by a SaM pipeline. We also considered for comparison the original J-Linkage without additional constraints and gap filling procedures. The results are shown in figure Fig. 6. It can be noticed how [13] fails to extract a consistent triangulation. The reason is that the simple subject of the scene is non-convex, and photo-consistency alone seems to be sufficiently powerful in this case. J-Linkage without constraints is able to correctly detect the supporting planes; yet, the final patches defined with a Delaunay triangulation contains gaps and fails to delineate the underlying object. Our approach obtains best results, even if some triangles are missing.

5.2 Real world examples

Three tests were performed on publicly available data[1] produced by the Structure and Motion pipeline described in [20]. Results show the fitting planes to the cloud of points, and the associated patches, projected over the images.

The first set - "Dante" - is composed of 39 images and 2971 points. The results are shown in Fig. 7. In the second test the subject is a church. The images involved are 54 and the cloud of points is composed of 11094 points. The results are shown in Fig. 8. The last test is computationally more challenging. The subject is "Piazza Bra" (Verona). The images are 380 and the points 52024 (obtained by subsampling the original 104047 points). The final extracted patches with our approach, visible in Fig. 9, are 302. It can be appraised from the examples shown as the patches are always covering planar regions of *actual* surfaces, whereas planes found by J-linkage not always correspond to a physical

[1] http://profs.sci.univr.it/~fusiello/demo/samantha/



(a) Triangulation produced by [13]. (b) Triangulation produced by [13]. (c) Final surface produced by [13].

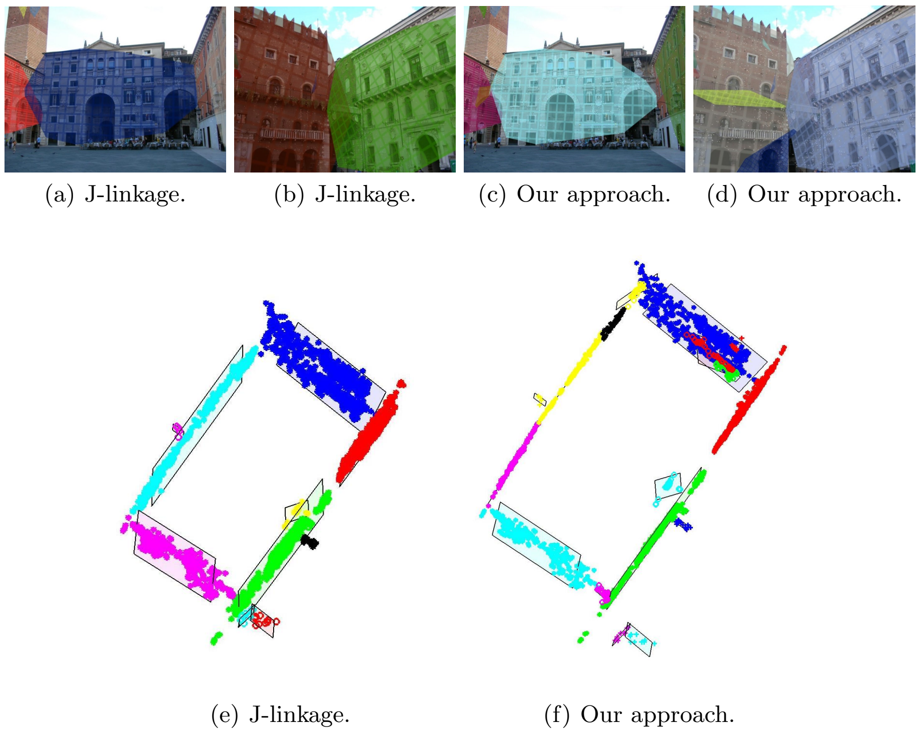
(d) Triangulation produced by J-Linkage. (e) Triangulation produced by J-Linkage. (f) Final surface produced by J-Linkage.

(g) Triangulation produced by our approach. (h) Triangulation produced by our approach. (i) Final surface produced by our approach.

Fig. 6. “Duplo” example. The top row depicts the results produced by [13], the middle row the results produced by J-linkage followed by a Delaunay triangulation and the bottom row shows the results of our approach.

plane (see for example the triangle in the sky of Fig. 8(a)). Please note that the boundaries of the patches seldom do not coincide with the actual edges of the façades, because points were detected by SIFT, which tends to keep away from corners. However, these planar patches must be considered only as a initial step toward the extraction of an high-level model. Several heuristics can be deployed to expand the regions up to their natural boundaries.

The code is entirely written in C++ and it is written upon J-Linkage [2]. The computing time on an entry level PC with a single core 2.4Ghz cpu, is about , 20 seconds for the "Duplo" example, 15 minutes for "Dante", 1 hour for "Pozzoveggiani" and 14 hours for "Piazza Bra".

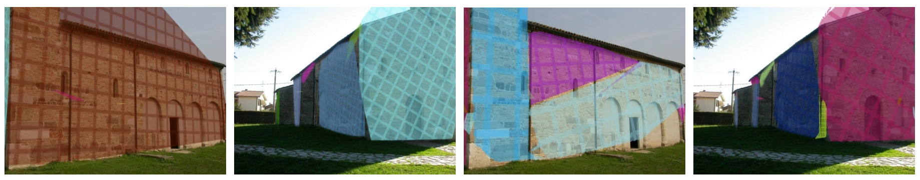

(a) J-linkage. (b) J-linkage. (c) Our approach. (d) Our approach.

(e) J-linkage. (f) Our approach.

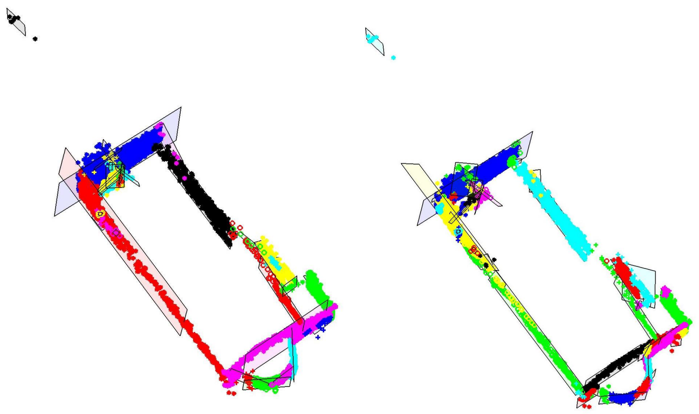
Fig. 7. "Dante" dataset. The top row (a-c) depicts the patches superimposed onto the images. The bottom row (e,f) shows the supporting planes from an azimuth view.

For visualization purposes only we produced a textured version of our results shown in Fig. 10. The procedure we followed is straightforward: For every patch we have defined an alpha blended textured quad. The quad coordinates are

[2] http://profs.sci.univr.it/~fusiello - http://www.toldo.info/roberto

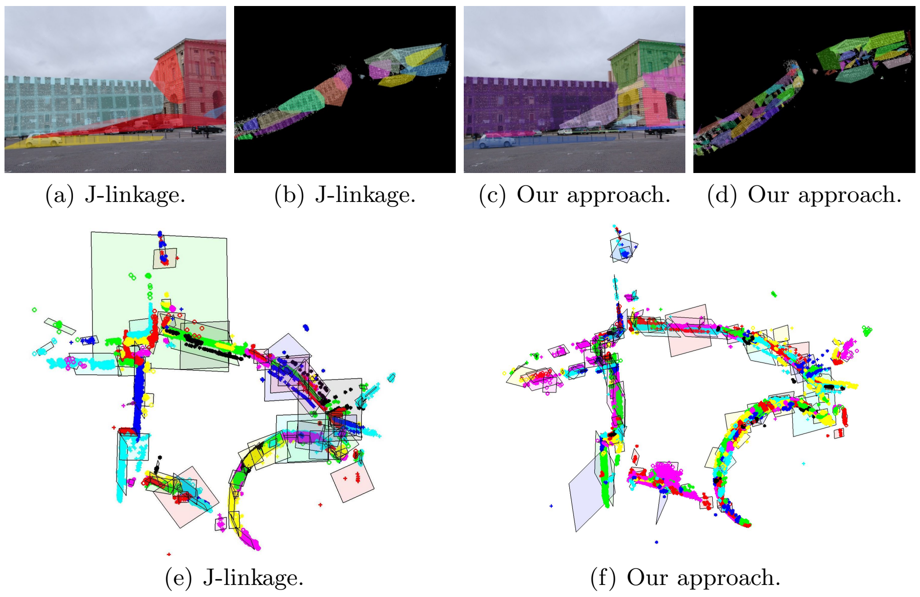


(a) J-linkage. (b) J-linkage. (c) Our approach. (d) Our approach.



(e) J-linkage. (f) Our approach.

Fig. 8. “Pozzoveggiani” dataset. The top row (a-c) depicts the patches superimposed onto the images. The bottom row (e,f) shows the supporting planes from an azimuth view.


(a) J-linkage. (b) J-linkage. (c) Our approach. (d) Our approach.

(e) J-linkage. (f) Our approach.

Fig. 9. "Piazza Bra" dataset. The top row (a-c) depicts the patches superimposed onto the images. The bottom row (e,f) shows the supporting planes from an azimuth view.

settled in order to include all the points of the patch projected on the supporting plane.

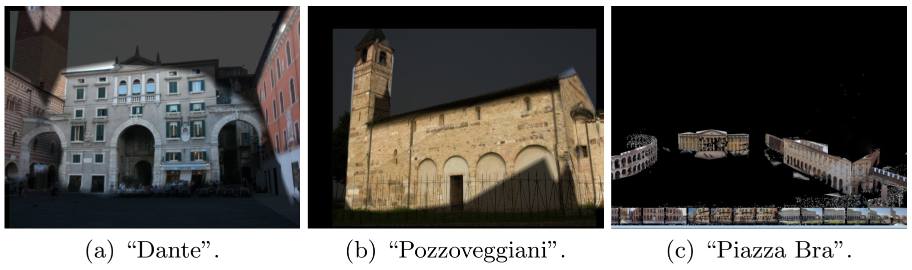

(a) "Dante". (b) "Pozzoveggiani". (c) "Piazza Bra".

Fig. 10. Textured examples.

6 Discussion

In this work we proposed a novel method for extracting planar photo-consistent patches that can cope with fairly large and noisy datasets coming from a standard SaM pipeline.

The spatial information has been seamlessly combined with the information coming from the images and the SaM pipeline. The final result is a very compact and stable intermediate representation, and can be regarded as a starting point for a complete automatic reconstruction of scene surfaces. Future work will aim at bridging further the semantic gap.

References

1. Taylor, C., Debevec, P., Malik, J.: Reconstructing polyhedral models of architectural scenes from photographs. Lecture Notes in Computer Science **1065** (1996) 659–670
2. Cipolla, R., Robertson, D., Boyer, E.: Photobuilder–3d models of architectural scenes from uncalibrated images. In: IEEE Int. Conf. on Multimedia Computing and Systems. Volume 1. (1999) 25–31
3. Wilczkowiak, M., Sturm, P., Boyer, E.: Using geometric constraints through parallelepipeds for calibration and 3D modeling. IEEE transactions on pattern analysis and machine intelligence (2005) 194–207
4. van den Hengel, A., Dick, A., Thorm
"ahlen, T., Ward, B., Torr, P.: VideoTrace: rapid interactive scene modelling from video. In: Proceedings of the SIGGRAPH conference. Volume 26., ACM New York, NY, USA (2007)
5. Schindler, K., Bauer, J.: A model-based method for building reconstruction. In: IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis. (2003) 74–82
6. Alegre, F., Dellaert, F.: A probabilistic approach to the semantic interpretation of building facades. Technical report, Georgia Institute of Technology (2004)
7. Dick, A., Torr, P., Cipolla, R.: Modelling and interpretation of architecture from several images. International Journal of Computer Vision **60** (2004) 111–134
8. Brenner, C., Ripperda, N.: Extraction of facades using rjM-CMC and constraint equations. Photogrammetric Computer Vision (2006) 155–160
9. Tiana, Y., Zhub, Q., Gerkea, M., Vosselmana, G.: Knowledge-Based Topological Reconstruction for Building Façade Surface Patches. In: 3D Virtual Reconstruction and Visualization of Complex Architectures (3D-ARCH). Volume 18. (2009)
10. Han, F., Zhu, S.: Bayesian reconstruction of 3d shapes and scenes from a single image. In: Workshop on High Level Knowledge in 3D Modeling and Motion. Volume 2. (2003)
11. Muller, P., Zeng, G., Wonka, P., Van Gool, L.: Image-based procedural modeling of facades. ACM Transactions on Graphics **26** (2007) 85
12. Hilton, A.: Scene modelling from sparse 3D data. Image and Vision Computing **23** (2005) 900–920
13. Morris, D., Kanade, T.: Image-consistent surface triangulation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Volume 1., IEEE Computer Society; 1999 (2000)
14. Nakatsuji, A., Sugaya, Y., Kanatani, K.: Optimizing a triangular mesh for shape reconstruction from images. IEICE Transactions on Information and Systems (2005) 2269–2276
15. Cooper, O., Campbell, N., Gibson, D.: Automatic augmentation and meshing of sparse 3D scene structure. In: Seventh IEEE Workshops on Application of Computer Vision. Volume 1. (2005)

16. Torr, P., Zisserman, A.: MLESAC: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding **78** (2000) 138–156
17. Bartoli, A.: A random sampling strategy for piecewise planar scene segmentation. Computer Vision and Image Understanding **105** (2007) 42–59
18. Moser, S., Wahl, R., Klein, R.: Out-of-Core Topologically constrained simplification for city modeling from digital surface models. In: 3D Virtual Reconstruction and Visualization of Complex Architectures (3D-ARCH). Volume 18. (2009)
19. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3D. In: SIGGRAPH: International Conference on Computer Graphics and Interactive Techniques. (2006) 835–846
20. Farenzena, M., Fusiello, A., Gherardi, R.: Structure-and-Motion Pipeline on a Hierarchical Cluster Tree. In: 3-D Digital Imaging and Modeling (3DIM). (2009)
21. Toldo, R., Fusiello, A.: Robust multiple structures estimation with J-linkage. In: Proceedings of the European Conference on Computer Vision (ECCV), Springer (2008) 537–547
22. Furukawa, Y., Curless, B., Seitz, S., Szeliski, R., Szeliski, R.: Towards Internet-scale Multi-view Stereo. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR). (2010) 12
23. Furukawa, Y., Curless, B., Seitz, S., Szeliski, R.: Reconstructing building interiors from images. In: Proceedings of the International Conference on Computer Vision (ICCV). (2009) 80–87
24. Myatt, D.R., Torr, P.H.S., Nasuto, S.J., Bishop, J.M., Craddock, R.: Napsac: High noise, high dimensional robust estimation - it's in the bag. In: British Machine Vision Conference. (2002)