Joint View Triangulation for Two Views

Maxime Lhuillier
Equipe MOVI, GRAVIR-IMAG and INRIA Rhône-Alpes,
ZIRST- 655 avenue de l'Europe,
38330 Montbonnot Saint-Martin, France.
Maxime.Lhuillier@inrialpes.fr

Abstract

We propose the *Joint View Triangulation*, which coherently models all visible and partially occluded patches within n views of a scene (rigid or not). It is built from an underlying dense matching and can be used for any application requiring discrete and efficient representation of deformation and displacement between views.

First robustness has to deal the unavoidable matching errors. Secondly matched and half occluded areas should be separated in each view to allow different processes on them. Finally, the elements of the structure which represent the matched area of each view pair should be in correspondence. This ensures a global *coherence* of the data and avoid redundant processes. In fact, we merely expect to an approximate but coherent structure, because of the finite precision of the images and bad matches.

This paper deals only with the two view case but also applies the joint view triangulation to morphing between real image pairs with large camera displacement.

Keywords: Constrained Delaunay Triangulation, Visibility, Morphing, Region Growing Matching.

1 Introduction

Motivations Many applications (e.g. image compression, image based rendering, layers, surface reconstructions, help for telemanipulation) need efficient rendering-oriented representations of a set of images. This paper introduces the joint view triangulation (JVT), a triangulation whose patches are shared between multiples views. It is an effective tool for modeling visibility information and improves existing solutions. It provides:

1. An image based representation with a reduced set of primitives, which approximates displacement maps.

2. For each view, a separation of matched and half occluded areas to allow different processes on them.

3. For each pair of views, a correspondence between primitives which represents the common (i.e. matched) areas of the pair. This ensures the global *coherence* of the data and avoids redundant processing during use.

Figure 1 shows an example of a JVT. A precise definition is given in the next section. Section 3 presents a robust algorithm for its construction in the two view case. Sections 4 and 5 apply it to the morphing of real image pairs, which are matched with a region-growing method. A report [13] describe the complete process (matching, JVT and warping).

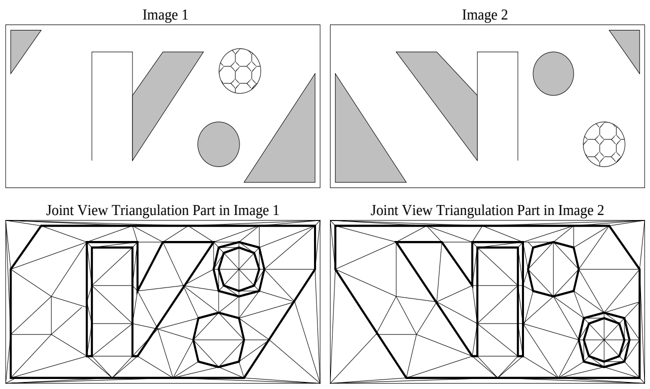
Image 1
Image 2
Joint View Triangulation Part in Image 1
Joint View Triangulation Part in Image 2

Figure 1: The first row represents two views of a non rigid scene, composed of a small vertical rectangle on an infinite horizontal plane and a falling ball. Half occluded areas (visible in only one image) are shaded gray. The second row shows a joint view triangulation for these two views. *Matched* (resp. *Unmatched*) *triangles* fill the matched (resp. unmatched) areas. The black edges represent the boundaries of matched areas and are forced to be edges of their respective triangulations.

Related Work Other structures have been proposed to model visibility information in computer vision and computer graphics (e.g. aspect graphs [6] and visibility skeleton [2]), but they need a rigid 3D model as input and are not optimized for the same uses. In contrast to these, our structure is directly constructed from a displacement map of a (possibly non rigid) scene.

A structure similar to the JVT was suggested as future work by [17] in the computer graphics field. To obtain a real-time visualization of a complex urban scene, they represent nearly objects as classical 3D models and distant scenery as 'impostors'. An impostor is a pre-calculated view of a model projected onto a transparent polygon, which is drawn instead of the model to accelerate the display process. The suggested problem was to generate a structure to obtain smooth transitions between two improved impostors.

Triangulation is often associated with the problem of surface reconstruction. In the rigid case, this problem is stronger than ours: A JVT can be deduced from a reconstructed surface simply by projecting the surface triangulation into the views. However, the JVT exists even for non-rigid scenes for which we have no 3D information

One class of surface algorithms generates triangulations from dense range images. For instance, [7] present a fast adaptative triangulation. An intermediate adaptative quadrilateral mesh is generated from the depth curvature, then the diagonal edges which best agree with the depth gradients and discontinuities are chosen. Another method [10] segments the range image using a surface orientation histogram and then spans each recovered smooth piece with independent triangulations. The discontinuities are thus preserved.

A second class of surface algorithms perturbs an initial surface so as to minimize the matching error. A robust method [5] combines diverse sources of information for the deformation: stereo and shape from shading data, 3D features and 2D silhouettes. A recent approach [4] guides a topology-variable surface using level set methods.

Our approach is closer to the adaptive triangulation method: A dense displacement mapping is converted to a JVT. However there are two differences: The matching is validated or invalidated during the conversion; and we generate triangulations that correctly treat the half occluded regions in the two views.

2 Joint View Triangulation for Two Views

Now, we define the JVT in the two view case (see the example in Figure 1), using the conditions 1,2 and 3 from the introduction.

2.1 Ideal Matching

We define the joint view triangulation for two views as a pair of inter related image triangulations, one for each image, based on an underlying locally dense displacement map. Triangulating in image space allows non rigid scenes to be handled. We call *matched triangle* (resp. *unmatched triangle*) a triangle which approximately covers a region of matched (resp. unmatched) pixels in its image. The Delaunay triangulation is chosen because of its good uniformity properties [15]. Matched and unmatched triangles are separated by constrained edges, which are forced to be part of the triangulation. If we assume that the displacement map is such that half occluded areas coincide with unmatched ones (ideal matching case), the condition 2 is satisfied. The *contours* are the sets of constrained edges which bound the sets of matched triangles in each image. We impose finally a one to one correspondence between each vertex (resp. edges of the contour) of different images to satisfy the condition 3.

2.2 Real Matching

In the real case however, matching methods do not produce matches in low textured areas. Thus we can not separate such areas from half occluded unmatched areas and condition 2 is violated. However this does not affect the coherence (condition 3) of our structure. A practical consequence is that there may be some unmatched triangles within a matched region. Some heuristic criteria to distinguish occlusions from untextured but unoccluded areas could be envisaged.

3 Algorithm

3.1 Overview

We propose a five-step algorithm which robustly converts an imperfect displacement map to a JVT.

- **Fitting**: Partition the first image into a set of independent regular patches, and for each one, try to fit a matched patch in the second image using inner matches (see subsection 3.2).
- **Averaging**: Remove small discontinuities and overlaps between the matched patches in the second image, by slightly moving their vertices to averaged locations. (see subsection 3.3).
- **Merging**: This step is more delicate. It grows the regions of matched triangles in the two images simultaneously in a coherent and robust way, by merging patches which can be smoothly joined to the current region boundaries (see subsection 3.4).
- **Completion**: The three previous steps are not optimal because of the parameter choices, but produce a coherent JVT. We improve the structure by declaring each unmatched triangle to be matched, if we can fit a matched patch to it. This step modifies the structure by swapping constrained-unconstrained status on existing edges and then it is easy.

- **Optimization**: This step depends on the application. One can improve the matching triangles accuracy by perturbing the vertices to optimize a criterion (e.g. correlation score, inlier rate, smoothness, epipolar errors...). Simplification is also possible by deleting some vertices. We have not used this step for the presented morphing application.

3.2 Fitting Step

Because of noisy and/or bad matches, some precaution is needed to obtain a reliable estimate of a matched patch P_2 in the second image from dense matches in a square patch P_1 in the first one (see Figure 2).

3.2.1 Principle

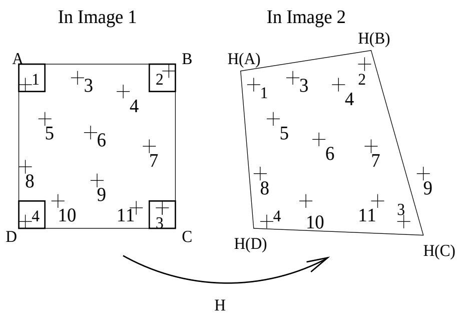

Figure 2: Points A,B,C, and D define a square patch P_1 in image 1. A sparse subset of the dense matching within is labeled from 1 to 11. The matches 1,2,3 and 4 (selected by a RANSAC trial) are respectively in the small framed neighborhood of vertices A,B,C and D. They are used to accurately define a planar homography H, which maps the square patch P_1 in image 1 to the distorted square patch P_2 defined by transformed points $H(A), H(B), H(C)$ and $H(D)$ in image 2. All matches (except 9) are compatible with H.

We try to fit a plane homography H (see the next subsection) using a RANSAC-like [3] procedure from the dense matches within P_1. If one is found, the relative coherence of the matches is checked and P_2 is defined by $P_2 = H(P_1)$. However, we do not accept the patch match (P_1, P_2) if the distortion of P_2 is too large.

For each RANSAC trial, four matches are selected in the square; this defines a trial homography. These four matches are chosen from the neighborhood of the four corners to obtain a usable accuracy and to ensure a good match distribution. The second part of a RANSAC trial counts the number of matches in the square compatible with the current homography. The best homography maximizes the number of inliers.

3.2.2 Plane Homography

A point in an image is represented by its homogeneous coordinates $(x, y, z)^\top$ or its Cartesian coordinates $(x/z, y/z)^\top$. A plane homography H is a one to one mapping which transforms a point $m_1 = (x_1, y_1, z_1)^\top$ to a point $m_2 = (x_2, y_2, z_2)^\top$ such that

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}.$$

Homogeneous coordinates and the homography matrix are defined up to a non zero scalar factor.

3.3 Averaging Step

The previous step provides a set of patch matches (P_1, P_2), but the patches in the second image are not exactly adjacent (see Figure 3). We next perturb the vertex locations in the second image to eliminate the small discontinuities and overlaps between patches.

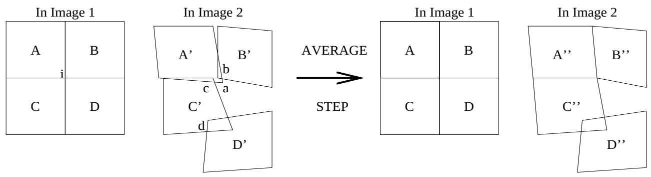

Figure 3: A,B,C and D are four patches of image 1 and A',B',C' and D' are their corresponding patches in image 2. The average step forces some patch vertices to coincide if they are enough close from each others. A", B", C", and D" are the result of the averaging step. Note that this step improves but can not solve all cases of intersecting patches (e.g. C" with D").

We do this by merging any of the four vertices (a, b, c and d) which are within a distance $s_{connect}$ of at least one other vertex, by averaging all vertices within this connected component. Note that this step improves but can not solve all cases of intersecting patches.

3.4 Merging Step

The previous steps produce a globally incoherent set of patch matches because of intersections. Many maximal and non-self intersecting subsets are possible. The merging step selects one of these and converts it to an incomplete but coherent JVT (the next step will complete it). The coherence between the two views is maintained at each stage of the merging step (i.e. a one to one correspondence between all matched vertices and contour edges in the two images).

3.4.1 Principle

The sets of matched triangles is grown simultaneously in the two images (see Figure 4) using the two operators: $InsertMatchedTriangle$ and $ForceMatchedQuadrangle$ (see subsection 3.4.3). Each checks the current triangulation to decide whether its operation is feasible, and if so greedily applies it. If it is not feasible, an unmatched gap (a set of unmatched triangles) is left in the final triangulation, unless the following completion step fills it. We specify the merging step after describing the two operators in subsection 3.4.3.

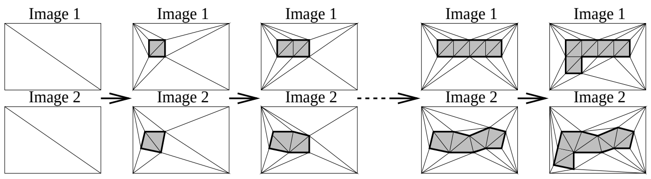

Figure 4: Gray (resp. white) triangles are matched (resp. unmatched) triangles. Black forced edges are constrained and form the contours. The sets of matched triangles grow simultaneously in the two images in a coherent way. Three insertions of patch matches (i.e. 3×2 $InsertMatchedTriangle$ operations) of a complete merging step are shown.

3.4.2 Topological Limitation

We chose to limit the possible topologies of the set of matched triangles to simplify our algorithm: We do not accept the cases shown in Figure 5. A first concrete consequence is that the contour (the polygonal boundary of the set of matched triangles) has a simple representation. Each of its vertices has only two links: the next and the previous vertex. A second consequence is that algorithms which use the structure are simplified too. For instance, a simple walk using edge adjacencies suffices to cover a complete connected component of the set of matched triangles.

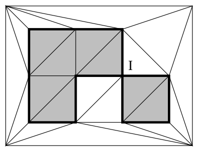

Figure 5: Gray (resp. white) triangles are matched (resp. unmatched). Black forced edges form the contour. More than two contour edges are adjacent to the same vertex I. We forbid a such case to simplify our algorithm and some others.

3.4.3 The operators $InsertMatchedTriangle$ and $ForceMatchedTriangle$

The $InsertMatchedTriangle$ operator takes the coordinates of three point matches in the two images and verifies that each of the three matches is *consistent* with the current structure. A new point match is *consistent* if its two points are corresponding vertices of a match of the contour, or are both outside of the set of matched triangles (see Figure 6). Second, it verifies that the resulting constrained edges would not intersect a contour in either image. The Figure 7 shows all good and some bad cases for edge configurations to obtain a success.

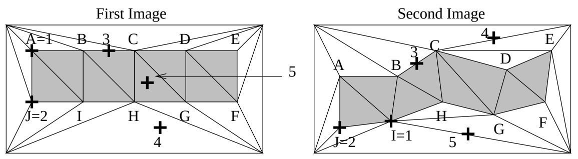

Figure 6: A,B...J are the matched vertices of gray and matched triangles in the two images. A new point match is *consistent* if its two points are corresponding vertices of a match of the contour, or are both outside of the set of matched triangles. Matches 2 and 4 are *consistent* but 1,3 and 5 are not. Note that match 4 is *consistent* even though the matching ordering constraint is violated (see the ball in Figure 1).

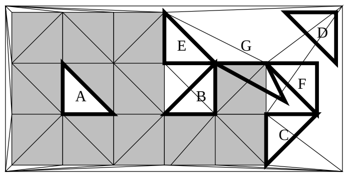

Figure 7: Gray triangles are matched triangles and the topology of their set is the same in the two images. Black framed triangles A,B,C and D (resp. E,F,G) are (resp. are not) correct input for the $InsertMatchedTriangle$ operator. Thus, triangle A,B,C can be added in both images to the current structure as matched triangles. Triangles E and F violate the topological restriction. One edge of Triangle G intersects an edge of the contour.

The $ForceMatchedQuadrangle$ operator takes coordinates of four point matches and verifies that each of the four matches coincides with a vertex match of the current structure. Second, it verifies that if the resulting constrained edges would not intersect a contour in either image. The Figure 8 shows the only three possible cases for a success. Note that we could not build a complete set of matched triangles homeomorphic to a planar ring without $ForceMatchedQuadrangle$.

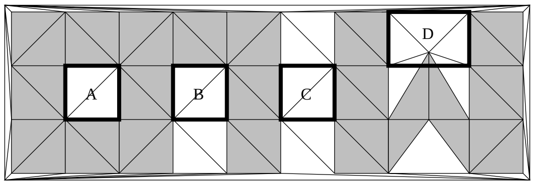

Figure 8: Gray triangles are matched triangles and the topology of their set is the same in the two images. Black framed quadrangles A,B,C (resp. D) are (resp. are not) correct input for the $ForceMatchedQuadrangle$ operator. Thus, Quadrangles A,B,C can be added in both images to the current structure as matched triangles. One edge of quadrangle D intersect an edge of the contour.

These two operators are used in the merging step. A maximal set of coherent patch matches is converted into two dependent sets of matched triangles in the two images as follows. For each patch line in the first image,

1. use $InsertMatchedTriangle$ operator twice for each patch match (see Figure 4).
2. complete some gaps in the current line by trying to grow the set of matched triangles using the operators $InsertMatchedTriangle$ and $ForceMatchedQuadrangle$.

4 Experimental Results

Results on real image pair are now discussed.

4.1 Input for Our Tests

We construct joint view triangulations from displacement maps obtained by a greedy region growing based dense matching algorithm [12] with pixel accuracy. At each step, a new match is picked from the set of current matches and is used to detect other matches in its neighborhood [14], [12]. This set grows from an initial set of sparse matches like Harris points [9] matched with correlation (see Figure 9 (a)). Region growing matchings are reasonably effective even without epipolar constraints or displacement bounds, but can not match in low textured areas (see Figure 9 (b)). It is important for the present application (morphing) to match image pairs with large displacements.

4.2 Results

Figure 9 shows the joint view triangulation (c) constructed from pixel matching (b). This example exhibits large unmatched areas: the sky (untextured) and half occluded areas behind the trunk. Owing to the good uniformity properties of the Delaunay triangulation, these areas are connected to the closest matched ones. For instance, small unmatched areas at occlusion borders are rounded. This seems to be a good choice for our application, because it limits the amount of triangle deformation during the morphing process.

The algorithm runs quickly, for instance, the 360×240 gray images are matched in 4s and the joint view triangulation is constructed in 3s on a Ultra Sparc 300Mhz. In practice, we fit two sizes of square patches (16×16 and 8×8) to obtain more matched patches and accelerate the process.

However, the precision at the occlusion borders is fixed by patch size: roughly 8 pixels. Note that some unmatched triangles occur inside matched areas because of incorrect dense matching.

5 Using Joint View Triangulation for Morphing

A joint view triangulation is built from image pair I_0, I_1. We now describe how to use it to interpolate intermediate images $I(\lambda), \lambda \in [0,1]$ between I_0 and I_1. Like other morphing techniques (e.g. [18], [1], [11]), it uses heuristics to combine shape interpolation and texture blending. Heuristics are necessary because:

- The initial matching is incomplete.
- No depth information is available (non rigid scenes are allowed).

Figure 10 shows the image obtained for $\lambda = 0.5$ from the joint view triangulation of the flower garden image pair. Many others examples are available at our web site.

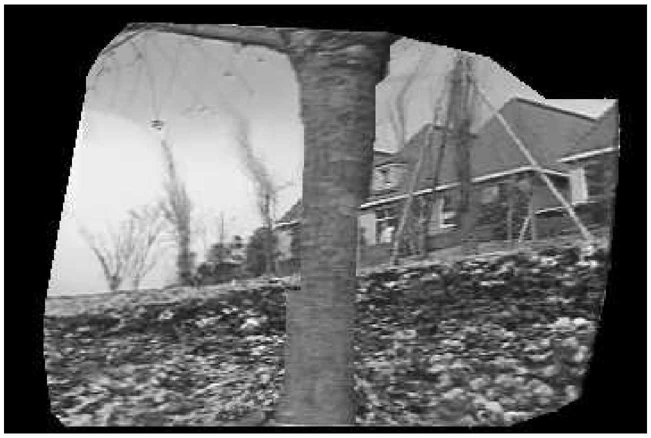

Figure 10: Morphing result ($\lambda = 0.5$) from the flower garden image pair using joint view triangulation (see the mpeg sequences at http://www.inrialpes.fr/movi/people/Lhuillie/demo.html).

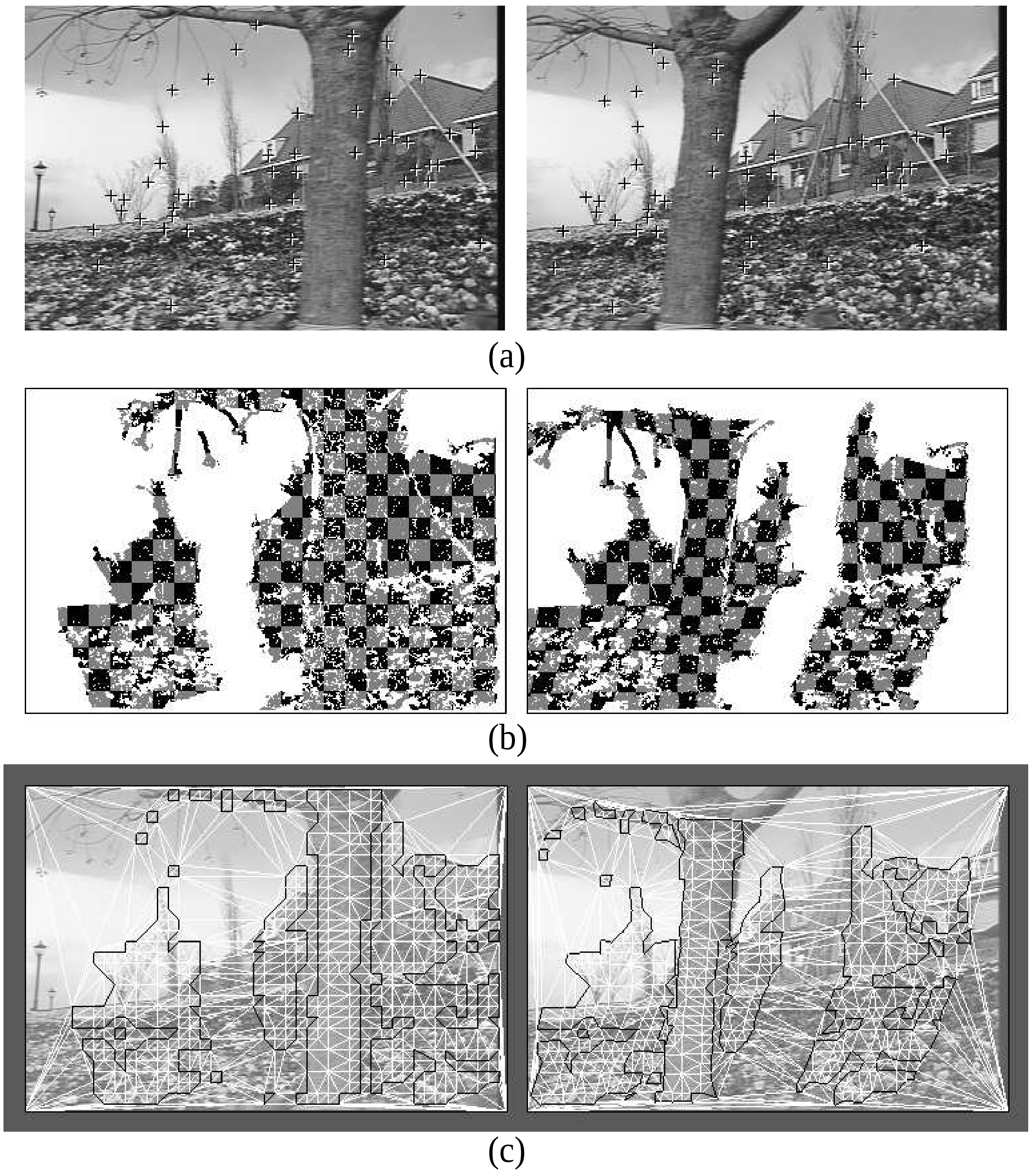

Figure 9: Flower garden image pair and initial sparse matches (a), region growing dense matching (b), the resulting joint view triangulation (c). Matched pixels of the left image (b) are colored with a gray-black checker-board, and the corresponding pixels of the right image are colored with the same color. This makes it easy to visualize the match of each square and its distortion. Black edges (c) are constrained and form the final contours of matched regions. All vertices and the contour edges are matched in the two views. White edges are Delaunay edges and are not necessarily matched.

5.1 Principle

We draw all triangles of the joint view triangulation except those which have an image corner as vertex. They are mapped using linear interpolation of their matched vertices. A modified version of the painters algorithm [8] is used to deal with variable depth components of the scene. The classic painters algorithm consists of drawing all triangles in a back-to-front order. Thus, the foreground is drawn over the previously drawn background.

Because non rigid scenes are allowed, we use a heuristic to obtain the ordering. We assume that the vertices with the largest displacements between I_0 and I_1 are the closest to the viewer. Note that this order is exact if the camera motion is a pure translation and the scene is rigid.

5.2 Heuristic

To obtain $I(\lambda)$ from I_0, I_1, let $\tilde{I}_0$ and $\tilde{I}_1$ be two intermediate buffers. First, all triangles of the joint view triangulation of I_0 (resp. I_1) are warped into $\tilde{I}_0$ (resp. $\tilde{I}_1$) in a painter-like order. The following subsection explains how to warp a single, matched or unmatched triangle. Next, the final image $I(\lambda)$ is obtained by texture blending of $\tilde{I}_0$ and $\tilde{I}_1$.

For the warp from I_0 to $\tilde{I}_0$, unmatched triangles are drawn before matched ones because they contain half occluded areas. The small unmatched triangles T_{nc} in I_0 with two vertices on different connected components (e.g. trunk and background) do not correspond to an object surface. Other unmatched triangles T_c which correspond to object surface (e.g. the flowers, tree and house which are half occluded by the trunk, matching negligences) should not be erased by triangles T_{nc} while drawing. We observe that T_{nc} triangles in I_0 implicitly define very stretched triangles in I_1 by vertex correspondences contrarily to triangles T_c. Thus, an heuristic order is defined from the corresponding vertices in I_1: We choose to draw all unmatched triangles v_0^0, v_1^0, v_2^0 such that the value $Max\{||v_0^1 - v_1^1||, ||v_1^1 - v_2^1||, ||v_2^1 - v_0^1||\}$ decreases, where v_i^1 is matched with v_i^0.

Matched triangles are then drawn. We assume that the vertices with the largest displacements between I_0 and I_1 are the closest to the viewer. Thus, we choose to draw all matched triangles v_0^0, v_1^0, v_2^0 in increasing order of $Max\{||v_0^1 - v_0^0||, ||v_1^1 - v_1^0||, ||v_2^1 - v_2^0||\}$.

For the final texture blending of $\tilde{I}_0$ and $\tilde{I}_1$, we use texture weights $s_0(x,y)$ and $s_1(x,y)$ for pixel (x,y) in $\tilde{I}_0$ and $\tilde{I}_1$, provided by the triangle warpings (see the next subsection). The resulting value of pixel $I(\lambda)(x,y)$ is then

$$I(\lambda)(x,y) = \frac{(1-\lambda)s_0(x,y)\tilde{I}_0(x,y) + \lambda s_1(x,y)\tilde{I}_1(x,y)}{(1-\lambda)s_0(x,y) + \lambda s_1(x,y)}.$$

5.3 Warp a Triangle

We have to draw a 'source' triangle with vertices $v_0^0, v_1^0, v_2^0 \in I_0$ (resp. $v_0^1, v_1^1, v_2^1 \in I_1$) of a joint view triangulation. Let v_0^1, v_1^1, v_2^1 (resp. v_0^0, v_1^0, v_2^0) be their respective vertex matches in the other image. The vertices $v_0(\lambda), v_1(\lambda), v_2(\lambda)$ of the 'destination' triangle in image $\tilde{I}_0$ (resp. $\tilde{I}_1$) are defined by λ: $v_i(\lambda) = (1-\lambda)v_i^0 + \lambda v_i^1$. The texture of the source triangle is mapped on the destination triangle using linear interpolation. We do not draw the source triangle if the destination is reversed

A texture weight s is also calculated for each pixel of the destination triangle. It is used to weight the final blending between $\tilde{I}_0$ and $\tilde{I}_1$. Normal weight ($s = 1$) is assigned for non stretched triangles, and low weight ($0 < s < 1$) for stretched ones. We use $s = Min(1, \frac{||v_0^0 v_1^0 v_2^0||}{||v_0^1 v_1^1 v_2^1||})$ (resp. $s = Min(1, \frac{||v_0^1 v_1^1 v_2^1||}{||v_0^0 v_1^0 v_2^0||})$ in our tests, where $||abc||$ is the area of the triangle a, b, c.

6 Conclusion and Future Work

We have proposed an image based structure called the joint view triangulation, which encodes in a coherent way all visible patches within views of a scene. A robust algorithm for its construction is presented for the two view case. An imperfect dense matching is converted into two corresponding constrained Delaunay triangulations by successive insertion of matched patches in both images. We have successfully used the structure for morphing between real image pairs with large camera displacements.

Some improvements are envisaged. First, a refinement of the matched region contour would increase the precision of the results. Second, variable sized patches would be a good way to represent previously matched regions like fine objects (e.g. electric-post) or large and untextured regions (e.g. manufactured object parts). Third, the underlying dense matching could be reconsidered and completed using the joint view triangulation (e.g. to match small and untextured areas).

ACKNOWLEDGMENTS Thanks to Long Quan, Roger Mohr and Bill Triggs for discussions and reading the paper. This work is partly supported by MENRT and PRA-AFCRST.

References

[1] T. Beier and S. Neely. Feature-Based Image Metamorphosis *SIGGRAPH'92 Proceeding*, pages 35–42, 1992.

[2] F. Durand, G. Drettakis and C. Puech. The Visibility Skeleton: A Powerful And Efficient Multi-Purpose

Global Visibility Tool. *Proceedings SIGGRAPH'97*, pages 89-100, 1997.

[3] M.A. Fischler and R.C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Graphics and Image Processing*, 24(6):381–395, 1981.

[4] O. Faugeras and R. Keriven. Complete Dense Stereovision Using Level Set Methods. *Proceedings of 5th European Conference on Computer Vision*, volume 1, pages 379–393, 1998.

[5] P. Fua and Y.G. Leclerc, Using 3-Dimensional Meshes To Combine Image-Based and Geometry-Based Constraints. *Proceedings of 4th European Conference on Computer Vision*, pages 281–291, 1994.

[6] Z. Gigus, J. Canny and R. Seidel. Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 13(6):542-551, 1991.

[7] M.A. Garcia, A.D. Sappa and L. Basafie. Efficient Approximation of Range Images Through Data-Dependent Adaptative Triangulation. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 628–633, 1997.

[8] J.D. Foley, A. van Dam, S.K. Feiner and J.F. Hughes. Computer Graphics: Principles and Practice. *Addison-Wesley*, 1991.

[9] C. Harris and M. Stephens. A combined Corner and Edge Detector. *Alvey Vision Conference*, pages 147–151, 1988.

[10] R. Koch. 3D Surface Reconstruction from Stereoscopic Image Sequences. *Proceedings of the 5th International Conference on Computer Vision*, pages 109–114, 1995.

[11] S.Y. Lee, K.Y. Chwa, J. Hahn and S.Y. Shin. Image Morphing Using Deformation Techniques. *The Journal of Visualization and Computer Animation*, volume 7, pages 3–26, 1996.

[12] M. Lhuillier. Efficient Dense Matching for Textured Scenes Using Region Growing. *the Ninth British Machine Vision Conference*, pages 700–709, UK, 1998.

[13] M. Lhuillier. Towards Automatic Interpolation for Real and Distant Image Pairs. Technical Report 3619, INRIA, February 1999 (demos and papers available at http://www.inrialpes.fr/movi/people/Lhuillie/demo.html).

[14] G.P. Otto and T.K. Chau. A region-growing algorithm for matching of terrain images. *Image and Vision Computing*, 7(2):83-94, 1989.

[15] D. Rippa. Minimal roughness property of the Delaunay triangulation. *Computer Aided Geometric Design*, volume 7, pages 489–497, 1990.

[16] Y. Shinagawa and T.L. Kunii. Unconstrained Automatic Image Matching Using Multiresolutional Critical-Point Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (D. Fellner and L. Szirmay-Kalos, eds.), 20(9):994-1008, 1998.

[17] F. Sillion,G. Drettakis and B. Bodelet. Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery. *Eurographics'97* , 16(3), 1997.

[18] G. Wolberg. Digital Image Warping. *IEEE Computer Society Press, Los Alamitos, California*, 1990.