MVA'98 IAPR Workshop on Machine Vision Applications, Nov. 17-19, 1998, Makuhari, Chiba, Japan

1—3

Randomized Adaptive Algorithms for Mosaicing Systems

Frank Nielsen*
SONY COMPUTER SCIENCE LABORATORY INC.

Abstract

Image-based rendering is a recent trendy area which aims to provide users with convincing photorealistic computer graphics in real time. In this paper, we mainly concentrate on randomized algorithms for finding the fundamental perspective matrix that relates two given pixelized images by applying Monte-Carlo randomized algorithms. Since the approach is rather generic, we derive an adaptive system which registers a sequence of images by choosing a trade-off between the generated image-matching quality and the time required to get an approximation of an ideal perspective transformation.

1 Introduction and motivations

Image-rendering systems are currently widely used since geometric models (surfaces, volumetric models, etc.) have been quite difficult to obtain from images. In this paper, we present a randomized heuristic for computing the perspective matrix that relates two images to each other. We describe some combinatorial geometric randomized methods based on photogrammetry which improves the running time of deterministic authoring systems. The method we propose below is fully automatic, does not require any human interactions and may be applied to high picture definition as well as movie compression.

2 What is mosaicing ?

Mosaicing consists in taking a sequence of still image pictures and provide a collection of transformations to join them into one blended picture. It is well-known [1] that in the case of images taken from a same three-dimensional viewpoint (with a pinhole-model camera), the transformation related these images are homographies (or collineations), defined up to a scalar factor, in the projective plane $\mathbb{P}^2$. Let $\mathbf{H}$ be such a transformation. Let p_1 and p_2 be the pixel points in image $\mathbf{I}_1$ and $\mathbf{I}_2$ that are the respective perspective projection of the same physical three-dimensional points p. Using the homogeneous coordinates for p_1 and p_2, we get:

$$p_2 = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \mathbf{H}p_1 = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix},$$

with $p_1 = (\frac{x_1}{z_1} \frac{y_1}{z_1})$ and $p_2 = (\frac{x_2}{z_2} \frac{y_2}{z_2})$.

There are mainly two widely used techniques that have been employed in the past: the first one consists in geometric registration where the collineation is often restricted to similitudes on the plane (and not a full perspective collineation). The second one is based on Fourier principles and is called the *phase correlation* method [2, 3]. Recently, Szeliski and Shum presented an elegant and robust method for creating full mosaics and texturing them onto a polyhedron [4].

3 General randomized techniques

We apply random sampling (see also [5, 6]) to algorithms that match features. We would like to find a perspective transformation that maps one picture $\mathbf{I}_1$ to another $\mathbf{I}_2$ in order to produce a single picture. If p_1 is the perspective projected point of the three-dimensional point p in image $\mathbf{I}_1$, and p is also visible (i.e., not obstructed) in picture image $\mathbf{I}_2$. Then $p_2 = \mathbf{H}p_1$, with $p_1 = (x_1\ y_1\ 1)^T$. Let us extract n_1 features (edges, corners, triple junctions, etc.) $\mathcal{S}_1$ from $\mathbf{I}_1$ and n_2 features $\mathcal{S}_2$ from $\mathbf{I}_2$. We aim at matching, by an appropriate collineation $\mathbf{H}$, as many as possible features. We initially attach to each detected feature a vector of characteristics describing the neighborhood where the feature has been extracted (e.g., intensity values, qualitative measures). We say that a point p_1 in $\mathcal{S}_1$ matched a point p_2 in $\mathcal{S}_2$ for a collineation $\mathbf{H}$ if $d_2(p_2, \mathbf{H}p_1) \leq \epsilon$ (for some given $\epsilon \geq 0$) and if their corresponding features correlate satisfactorily. Let $\mathbf{H}\mathcal{S}_1$ be the set of points $\{\mathbf{H}p | p \in \mathcal{S}_1\}$. We distinguish between three families of matchings:

Pixel Cross-Corellation. We use as a scoring

*Address: 3-14-13 Higashi-Gotanda, Shinagawa, Tokyo 141-0022, Japan. E-mail : nielsen@csl.sony.co.jp

function for $\mathbf{H}$ the zero mean cross-correlation on subwindows centered at extracted points.

The Haussdorf matching. Each point $p_2 \in \mathcal{S}_2$ is associated to its closest neighbor of $\mathbf{H}\mathcal{S}_1$. Eventually *several points* of $\mathcal{S}_2$ may be attached to a same point of $\mathbf{H}\mathcal{S}_1$. Each associated pair of points defines an edge of the graph whose nodes are the point set $\mathbf{H}\mathcal{S}_1 \cup \mathcal{S}_2$. The score of a Haussdorf matching is set to be the maximum edge length of any pair of matched points. (See also the directed partial Haussdorf distance.)

The bottleneck matching. The bottleneck measure [7, 8] reflects in a better way the similarity of point sets. We consider the complete bipartite graph $\mathcal{G} = (\mathbf{H}\mathcal{S}_1, \mathcal{S}_2, \mathcal{E})$, where $\mathcal{E}$ is the set of all weighted edges $e = (p_1, p_2)$ where $w(e) = d_2(p_1, p_2)$. Let $\mathcal{G}_\epsilon$ be the restricted bipartite graph to all edges of weights less than ϵ. The bottleneck matching is a maximum matching, often not perfect matching, of $\mathcal{G}_\epsilon$.

Let $m = |\mathcal{E}_\epsilon|$ be the number of edges of $\mathcal{G}_\epsilon$ and $n = n_1 + n_2$ be the number of vertices. Perfect/maximum matchings in general weighted graph, where one wants to minimize the sum of matched edges, require $O(n^3)$ time [9] using the so-called Hungarian method. On the other hand, on unweighted bipartite graphs, a maximum matching can be found whenever it exists in time $O(m\sqrt{n})$ [10]. When considering geometric graphs, i.e., graphs obtained from a geometric scene, Vaidya [11] gave an $O(n^{\frac{5}{2}})$-time algorithm for matching two points sets so that the sum of the matched edges is minimized. Considering the L_∞ distance instead of the L_2 distance, Vaidya obtained an $O(n^2 \log^3 n)$-time algorithm. Later on, those results where improved by Agarwal et al. [12] to $O(n^{2+\gamma})$ for any arbitrary small positive $\gamma > 0$. Very recently, Efrat and Itai [13] using an implicit form of the geometric graph reported nearly $O(n^{\frac{3}{2}})$-time algorithm for computing a longuest matching that minimizes the maximum edge length among all matched edges. Further refinements of their algorithm has been achieved by using dynamic data-structures for fat objects [19]. (See also the work of Heffernan and Schirra [14] for approximation schemes.) Another classic approach to pattern matching, first developed by Huttenlocher et al. [15], is to perform a *branch-and-bound strategy* on some search space; For example, the coefficients of the matrix coding an affine transformation, define a 6-dimensional space. The algorithms start with a given box containing an optimal solution, splits the current box into subboxes, kill some of them (those where the current best solution is better than any solution provided by them) and branch on the remaining active subboxes. The process stops whenever an "acceptable" solution is found (depends on the required precision). Very recently, those methods have been extended using alignment as in Mount et al. [16] and, Hagedoorn and Veltkamp [17]. Our algorithm extends the combinatorial Monte-Carlo approach using properties of the Euclidean/projective space of point features.

Let α be the percentage of points required to match up to an absolute error ϵ (that is $\max\{\lceil \alpha |\mathcal{S}_1| \rceil, \lceil \alpha |\mathcal{S}_2| \rceil\}$ points at least). Parameter α is useful in practice since it reflects the perspective distribution of common feature points and possibly occluding parts (hidden features). Let k be the number of features required in $\mathcal{S}_1$ and $\mathcal{S}_2$ for computing a basic transformation that perfectly matches pair by pair these $2k$ features (edges, corners, triple junctions, etc.). Since we know that a significant portion of points in $\mathcal{S}_1$ will ϵ-match, choosing at random a k-tuple $\mathcal{P}_1$ and computing all induced homographies with all other k-tuples of $\mathcal{S}_2$ will lead to the more efficient (by a square root factor) Monte-Carlo algorithm. The originality of our method consists in filtering the potential tuples $\mathcal{P}_2$ by considering metric constraints imposed by the selection of $\mathcal{P}_1$. We call it *geometric filtering* and it allows both in practice and theory to speed up the algorithm. For sake of simplicity, let us assume that we look for a translation and a rotation matching features of $\mathbf{I}_1$ into $\mathbf{I}_2$. Each detected feature point p lies in a ball $\mathcal{B}(p, \mu)$. We set $\epsilon = 4\mu$ in order to take into account the fuzziness of our features. Let p^* denote the "visual" feature point that our feature extraction algorithm approaches to p ($p^* \in \mathcal{B}(p, \mu)$). Between two feature points, we have $|d_2(p_1^*, p_2^*) - d_2(p_1, p_2)| \leq 2\mu$. Assuming some knowledge of the uniform scaling factor related the two images, instead of comparing (L_1, L_2) to all pairs (R_1, R_2), we first choose a point $R_1 \in \mathbf{I}_2$ and choose the second point inside the ring whose center is R_1 with minimum circle $\mathcal{B}(R_1, d_2(L_1, L_2) - 2\mu)$ and width 4μ. In the ideal case where the feature extraction algorithm ensures that $p = p^*$ for all features, we avoid testing all the $\binom{n_2}{2}$ pairs. Indeed, the maximum number of "unit" distances defined by a collection of points is in between $\Omega(n^{1+\frac{c}{\log\log n}})$ and $O(n^{\frac{4}{3}})$, where c is a constant. (Proving $O(n^{\frac{3}{2}})$ is easy by decomposing a circle query into four monotone arc queries.) Erdös conjectured that the $\Omega(n^{1+\frac{c}{\log\log n}})$ bound is tight. This is related to the self similarity of a point set. We notice that we avoid testing most of the pairs (for example with $n_1 \simeq n_2 = 2000$ we skip more than 99% of the pairs). Geometric filtering is rather a general paradigm than some queries on rings. We may therefore extend this approach by tailoring it to the space of transformation anf feature types.

α/k	1	2	3	4	5
0.15	6.1	55	327.9	1199.3	13238.3
0.20	7.2	26.4	77.1	689.6	18586.1
0.35	2.9	8.4	19	59.5	321.4
0.5	2.1	3.7	7.8	25.6	30.7
0.60	1.7	2.7	5	12.2	14
0.75	1.4	2	2.2	3.5	4.2

Table 1: Number of times, l, that the program enters the `while loop` before finding a good tuple ($1 \leq k \leq 5$) which defines an (α, ϵ)-match.

The other natural way to randomize, as already noticed by Irani and Raghavan [6] (for alignment and geometric hashing problems), is to avoid to test the entire set $\mathcal{S}_2$ but rather to test only a subset of it, of size r. One way to proceed, is to select at the very beginning a subset $\mathcal{S}_1'$ of $\mathcal{S}_1$ of size r and for each rigid perspective transformation tests whether λr points of $\mathcal{S}_1'$ matches $\mathcal{S}_2$ (thru $\mathbf{H}$). If there exists such a transformation that provides λr matching points, we test all of the n_1 points. Notice that the subset $\mathcal{S}_1'$ may match locally several times in $\mathcal{S}_2$. We will denote by s the maximum number of times that $\mathcal{S}_1'$ can match in $\mathcal{S}_2$. That is s accounts for the maximum number of distinct collineations that produces a (λ, ϵ)-match of $\mathcal{S}_1'$. In practice, s is very small and relates to the self-similarity [6] of a point set achievable by some families of transformations.

1: $\mathbf{H} = Id$
2: **while** not found a (α, ϵ)-matching homography $\mathbf{H}$ **do**
3: Choose r points $\mathcal{S}_1'$ from $\mathcal{S}_1$
4: Draw at random a k-tuple $\mathcal{P}_1$ from $\mathcal{S}_1'$
5: **while** not STOP **do**
6: Draw at random a k-tuple $\mathcal{P}_2$: $k = |\mathcal{P}_2|$ points of $\mathcal{S}_2$
7: (* We use **geometric filtering** *)
8: **for all** permutations $\mathcal{P}_2'$ or $\mathcal{P}_2$ **do**
9: Compute the homography $\mathbf{H}$ that perfectly matches tuple $\mathcal{P}_1$ to tuple $\mathcal{P}_2'$
10: **if** $\mathcal{S}_1'$ is a (λ, ϵ)-match in $\mathcal{S}_2$ **then**
11: **if** $\mathcal{S}_1$ is a (α, ϵ)-match in $\mathcal{S}_2$ **then**
12: **if** the characteristics of matched points correlate satisfactorily **then**
13: STOP
14: **end if**
15: **end if**
16: **end if**
17: **end for**
18: **end while**
19: **end while**

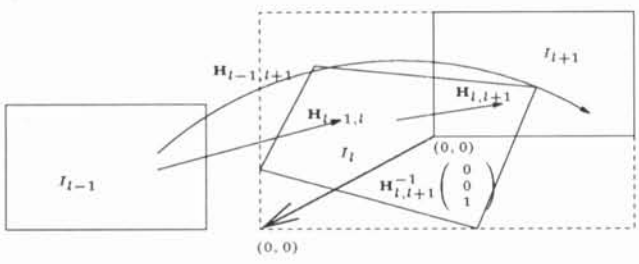
Putting it all together, we get the probability of failure after the ith iteration as follows:

$$\Pr(\textit{Failure}) \leq \Big((1-\alpha^k) + \exp(-\alpha(1-\lambda)r)\Big)^i.$$

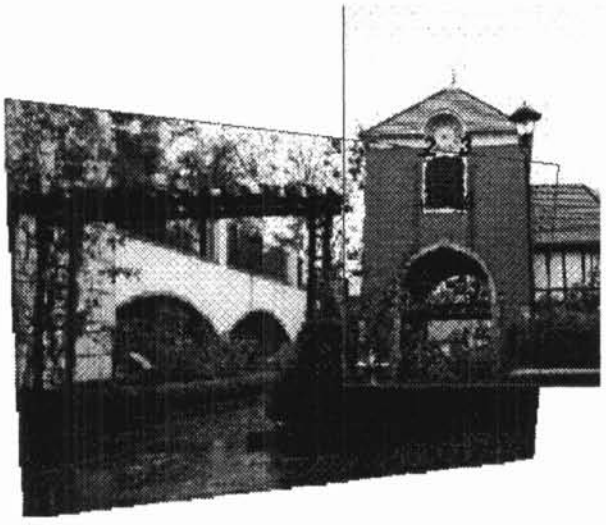
As a corollary, for any $\alpha > 0$, we can appropriately choose λ, r and i such that the algorithm will fail with probability at most $\frac{1}{2^q}$ by only multiplying the overall cost by $O_{\lambda,\alpha,r,k}(q)$.

We can implement easily the Haussdorf matching as follows: we first build a Voronoi diagram on points of $\mathcal{S}_2$ in $O(n_2 \log n_2)$ time [18]. Then, whenever we want to check for a (α, ϵ)-matching, we make n_1 queries inside the Voronoi diagram for a total cost of $O(n_1 \log n_2)$. Therefore, under the Haussdorf matching, it costs $O(\mathcal{F}_k(n_2) r \log n_2 + s n_1 \log n_2)$, where both s and $\mathcal{F}_k(n_2)$ are less than $k!\binom{n_2}{k}$. Implementing the bottleneck matching is more costly. Efrat and Itai [8, 19] proposed an interative $O(n_2 \log n_2 + n_1^{\frac{3}{2}} \log n_2)$-time algorithm to determine whether there is an (α, ϵ)-matching or not. Using their algorithm as the test procedure, we obtain an $O\left(\mathcal{F}_k(n_2)(n_2 + r^{\frac{3}{2}}) \log n_2 + s n_1^{\frac{3}{2}} \log n_2\right)$-time randomized algorithm. Our implemented system is adaptive by first looking for a rough translation ($k = 1$, a *anchor point*). This will allow us to subdivide images and therefore increase the overlapping ratio of some parts of them. We then detect more precise features and find progressively a better collineation (with $k = 4$ at the final stage). We also perform subpixel analysis and perturbation methods in order to improve the quality of the final picture.

Given a sequence of pictures taken from a handheld camera, we may create a poster by composing matrices as depicted in the following figure:



The software has been written in C++ and currently weights about 10K lines of code.



Acknowledgments

I would like to express my gratitude to Imad Zoghlami [20] (INRIA Robotvis – France) who introduced me to mosaicing systems and to Pr. Mario Tokoro (Sony CSL Inc. – Japan) for supporting my project researchs.

References

[1] Olivier Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, Cambridge, Massachusetts, 1993.

[2] C. D. Kuglin and D. C. Hines. The phase correlation image alignment method. In *IEEE Int. Conf. on Cybernetics and Society.*, pages 163–165, 1975.

[3] James Davis and Clay Kunz. Image mosaics via phase correlation. In *manuscript of Stanford University*, 1998.

[4] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic mosaics and environment maps. *SIGGRAPH 97*, pages 251–258. ACM SIGGRAPH, August 1997.

[5] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[6] Sandy Irani and Prabhakar Raghavan. Combinatorial and experimental results for randomized point matching algorithms. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 68–77, 1996.

[7] D. S. Hochbaum and D. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, 33:533–550, 1986.

[8] Alon Efrat and Alon Itai. Improvements on bottleneck matching and related problems using geometry. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 301–310, 1996.

[9] Harold W. Kuhn. *On the Origin of the Hungarian Method*, chapter 8. Elsevier Science Publishers B. V., Amsterdam, 1991.

[10] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing, Springer Verlag (Heidelberg, FRG and NewYork NY, USA)-Verlag*, 2, 1973.

[11] Pravin M. Vaidya. Geometry helps in matching (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 422–425, Chicago, Illinois, 2–4 May 1988.

[12] P. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. In *Proceedings of the 11th Annual Symposium on Computational Geometry*, pages 39–50, New York, NY, USA, June 1995. ACM Press.

[13] Alon Efrat and Alon Itai. Improvements on bottleneck matching and related problems using geometry. In *Proceedings of the Twelfth Annual Symposium On Computational Geometry (ISG '96)*, pages 301–310, New York, May 1996. ACM Press.

[14] Heffernan and Schirra. Approximate decision algorithms for point set congruence. *CGTA: Computational Geometry: Theory and Applications*, 4, 1994.

[15] D. P. Huttenlocher, G. A. Klanderman, and W. J. Ruclidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15, 1993.

[16] D. Mount, N. Netanyahu, and J. Le Moigne. Improved algorithms for robust point pattern matching and applications to image registration. In *14th Annual ACM Symposium on Computational Geometry*, 1998.

[17] M. Hagedoorn and R. Veltkamp. A general method for partial point set matching. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 406–408, 1997.

[18] J.D. Boissonnat and M. Yvinec. *Algorithmic geometry.* Cambridge university press, 1998.

[19] A. Efrat, M. J. Katz, F. Nielsen, and Micha Sharir. Dynamic data structures for fat objects and their applications. In *Proc. 5th Workshop Algorithms Data Struct.*, pages 297–306, 1997.

[20] I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2d mosaic from a set of image. In *Computer Vision and Pattern Recognition'1997*, 1997.