PAPER *Special Issue on Machine Vision Applications*

Randomized Adaptive Algorithms for Mosaicing Systems*

Frank NIELSEN[†], *Nonmember*

SUMMARY Given a set of still images taken from a hand-held camera, we present a fast method for *mosaicing* them into a single blended picture. We design time- and memory- efficient still image mosaicing algorithms based on geometric point feature matchings that can handle both arbitrary rotations and large zoom factors. We discuss extensions of the methodology to related problems like the recovering of the epipolar geometry for 3d reconstruction and object recognition tasks.

key words: image processing, registration, warping , mosaicing, point pattern matching, bucketting

1. Introduction

Mosaicing (also called *image stitching*) consists of taking a sequence of still image pictures and providing a collection of transformations to join/merge them into one blended picture (see Figure 1 and 9). Many software companies already propose stitchers for creating panoramas and browsers for navigating through them via environment maps (image-based rendering systems). Basically, those programs proceed as follows: (1) find or ask for camera parameters, (2) warp images according to these parameters (lines bend to quadratic curves) and (3) stitch images by means of a 1d- or 2d-translation and eventually small tilting, and (4) adjust and blend color intensities. Panoramic images do not preserve lines. We refer the reader to the course note [2] for an up-to-date survey on image registration and image warping techniques.

In this paper, we consider *perspective mosaicing* where straightness of lines is preserved. Our method is *automatic* and does not assume any user input nor any *a priori* knowledge of the camera parameters. Images can also be taken by several pinhole cameras having different intrinsic/extrinsic parameters. In the case of images taken by an ideal pinhole-model camera from (a) the same three-dimensional viewpoint or (b) a planar surface taken from two different viewpoints, the transformations related these images are known to be homographies (also called collineations [3]), linear transformations defined up to a scalar factor, in the projective plane $\mathfrak{P}^2$. A key difference from panoramic mosaicing is that we can interpret the composite stitching as the image that would have been taken by a bigger sensing device (e.g., CCD).

There are two widely used techniques that have been used so far in the past for mosaicing: the first one consists of *image registration* where the collineation is often restricted to similitudes on the plane (*id.* translation, rotation and scaling). Local image registration is usually either performed by gradient descent or Levenberg-Marquadt optimization procedure. Global image registration is reached (sometimes) by hierarchical matching using pyramidal image representation. The second method is based on Fourier principles and is called the *phase correlation* method. This technique estimates a global 2d translation by computing the phase differences of the respective image frequency signals but loses the perspective information as it goes to the frequency domain. Moreover for large rotations or different zoom scales, this method fails. Szeliski and Shum presented an elegant and robust method for creating full mosaics and texturing them onto a polyhedron [4], given rough image matchings.

Our proposed fully automatic method handles large zoom ($\simeq \times 3$) and arbitrary rotation values while keeping the running time attractive for responsive applications. Its current limitations, as discussed in section 6, are mostly due to the detection of *reliable* features (sometimes called "repeatable" or "stable" features) in images having large different scalings rather than the matching process in itself. Indeed, our method is based on planar point set pattern matching having a given precision tolerance. As the zoom range grows, say linearly, the tolerance window needs more than linear growth because of the detected feature imprecisions. Our method is based on Monte-Carlo/Las Vegas algorithms that combine both randomization and geometric feature selection.

Mosaicing is a core technology used in composite scanning, image-based rendering (e.g., new view generation without having fine 3d model description), high-definition pictures, image compression, image stabilization, etc. Still image mosaicing techniques largely differ from video mosaicing techniques [5] because no video stream (optical flow) is given for tracking *locally* points of interest in a prescribed window. In this paper, we concentrate our effort on finding those *global* feature

Manuscript received 1st October 1999
Manuscript revised 11th January 2000
[†]Sony Computer Science Laboratories Inc, FRL
*This paper was presented at MVA'98, Pages 11-14, ISBN 4-901122-98-3 (also available as technical report SCSL-TR-98-035 — see [1])

correspondences efficiently.

2. Feature-based mosaicing

Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be two pictures taken from the same optical center. We first start by detecting geometric feature (points, edges, triple junctions, etc.) sets $\mathcal{S}_1 = \{L_1, ..., L_{n_1}\}$ ($n_1 = |\mathcal{S}_1|$) and $\mathcal{S}_2 = \{R_1, ..., R_{n_2}\}$ ($n_2 = |\mathcal{S}_2|$). For example, points are extracted by a Harris-Stephens corner detector or even more reliably (at the expense of a small increase of the running time) by a Kanade-Lucas-Tomasi's intensity gradient approach. A collineation or homography is defined by the pairing $L_i \leftrightarrow R_i$ of 4 points $\{L_i\}_i$ of $\mathcal{S}_1$ with 4 other points $\{R_i\}_i$ of $\mathcal{S}_2$ in general position. Let $\mathbf{H}$ be such a transformation. Let L_i and Q_i 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 M_i. Using the homogeneous 2d coordinates for L_i and R_i, we get:

$$R_i = \begin{pmatrix} x'_i \\ y'_i \\ z'_i \end{pmatrix} = \mathbf{H}L_i = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

with $L_i = (\frac{x_i}{z_i} \frac{y_i}{z_i})$ and $R_i = (\frac{x'_i}{z'_i} \frac{y'_i}{z'_i})$. Let $\mathbf{P} = (L_1^T L_2^T L_3^T L_4^T)$ and $\mathbf{Q} = (R_1^T R_2^T R_3^T R_4^T)$ be 3x4 matrices. Then, we get $\mathbf{H} = \mathbf{Q}\mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}$.

Fig. 1 Mosaicing of 4 images (Stanford University).

Considering only homographies defined by feature correspondences, we have as many as $4!\binom{n_1}{4}\binom{n_2}{4}$ possible transformations (Example: for $n_1 = n_2 = 40$ we have more than 200 billion induced homographies. In comparison, we have only about 1.2 millions panoramic transformations). The naive algorithm which consists in scoring one-by-one each homography and reporting one which has the best score is therefore not scaleable ($O(n^8)$ homographies, where $n = \max\{n_1, n_2\}$).

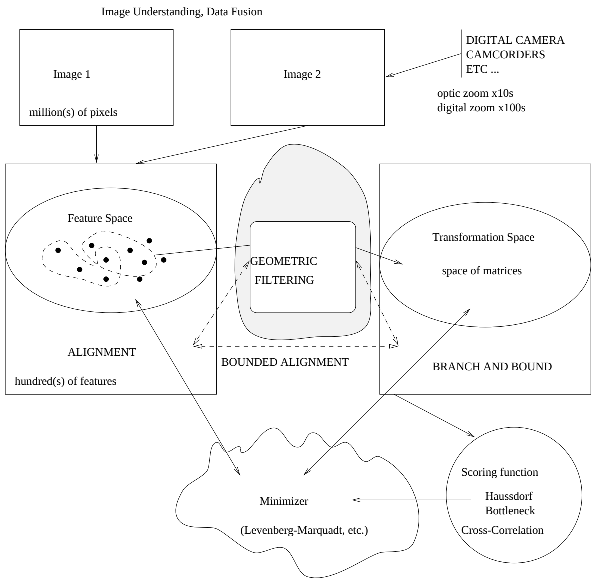

Fig. 2 Registration process synopsis: Geometric filtering selects potential feature candidates (feature space) that generate plausible transformations, i.e. reasonable zoom values, not too large rotations, etc. (transformation space)

If we consider anchored corners (a corner point and two half-line segments emanating from it) as features [6], the homography can be found by pairing only two anchored corners. Although the running time increases for detecting these elaborated templated features by non-linear minimization techniques, the core bottleneck is still the $O(n^4)$ combinatorics of the *all-pairing approach*. In this paper, we present an efficient scheme to select only a few (homographies) of them that are plausible to give birth to potential solutions.

The outline of the proposed algorithm is as follows (see also Figure 2):

- Extract features from images (e.g. corners).
- Give a set $\mathcal{T}$ of homographies satisfying *geometric constraints* and matching at least a given fraction of the point sets.
- Score each homography of $\mathcal{T}$ and choose one with highest quality (eg., score).

Once this global matching has been found, we refine the image quality by the following process:

- Perform local optimization (subpixel analysis) on features. This step is required because we deal with pixels and not points! We can use also a gradient descent or Levenberg-Marquadt method on pixel intensities.
- Perform local optimization by perturbating coefficients of the homography matrix (compensate for lens aberration).
- Warp images (deghosting techniques and pixel amplification).
- Correct intensity and blend images.

3. Scoring transformations

Once a transformation $\mathbf{H}$ has been chosen as a potential candidate ($\mathbf{H}$ is induced by 4 correspondence pairs), we have to evaluate its score. 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 L_1 in $\mathcal{S}_1$ matches a point R_1 in $\mathcal{S}_2$ for a collineation $\mathbf{H}$ if $d_2(R_1, \mathbf{H}L_1) \leqq \epsilon$ (for some prescribed $\epsilon \geqq 0$) and if their corresponding qualitative features correlate satisfactorily. We call this match an ϵ-match. Let $\mathbf{H}\mathcal{S}_1$ be the set of points $\{\mathbf{H}L | L \in \mathcal{S}_1\}$. The common feature points are called *inliers* and features present in only one of the two images are called *outliers*.

We distinguish between four families of scorings that exhibit trade-offs between running time and reliability of their scores:

(1) Pixel Cross-Corellation. We use as a scoring function for $\mathbf{H}$ the zero mean cross-correlation on subwindows centered at extracted points. On RGB pictures, a pixel p with color triple (p_R, p_G, p_B) has intensity $I(p) = 0.3p_R + 0.59p_G + 0.11p_B$. The quality is defined as follows:

$$Q(\mathbf{H}) = \frac{\sum_{p \in \mathcal{W}} (I(\mathbf{H}p) - I(p))^2}{\sqrt{\sum_{p \in \mathcal{W}} I(\mathbf{H}p)} \sqrt{\sum_{p \in \mathcal{W}} I(p)}},$$

where $\mathcal{W}$ is the correlation window.

(2) The Haussdorf matching. Each point $R \in \mathcal{S}_2$ is associated to its closest neighbor of $\mathbf{H}\mathcal{S}_1$ provided that its distance is less than a prescribed ϵ (see Figure 3). Eventually *several points* of $\mathcal{S}_2$ may be attached to the same point of $\mathbf{H}\mathcal{S}_1$ (especially when zoom factors differ). Each associated pair of points defines an edge of the graph whose nodes are the vertex set $\mathbf{H}\mathcal{S}_1 \cup \mathcal{S}_2$ (See Figure 3, top). The score of a Haussdorf matching is set to be the weigthed number of edges of any pair of matched points. (See also the directed partial Haussdorf distance that takes into account inliers/outliers.) This measure, however, is not suitable for different image scalings.

(3) The bottleneck matching. The bottleneck measure [7], [8] reflects in a better way the $1 \leftrightarrow 1$ matching 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 $e = (L, R)$ where $w(e) = d_2(L, R)$. Let $\mathcal{G}_\epsilon$ be the restricted bipartite graph of $\mathcal{G}$ containing all edges of weights less than ϵ: $\mathcal{G}_\epsilon = (\mathbf{H}\mathcal{S}_1, \mathcal{S}_2, \mathcal{E}_\epsilon)$, st. $\mathcal{E}_\epsilon = \{(\mathbf{H}L_i, R_j) | L_i \in \mathcal{S}_1, R_j \in \mathcal{S}_2, d_2(\mathbf{H}L_i, R_j) \leqq \epsilon\}$. The bottleneck matching is a maximum matching[†], often not perfect matching, of $\mathcal{G}_\epsilon$ (see Figure 3, bottom).

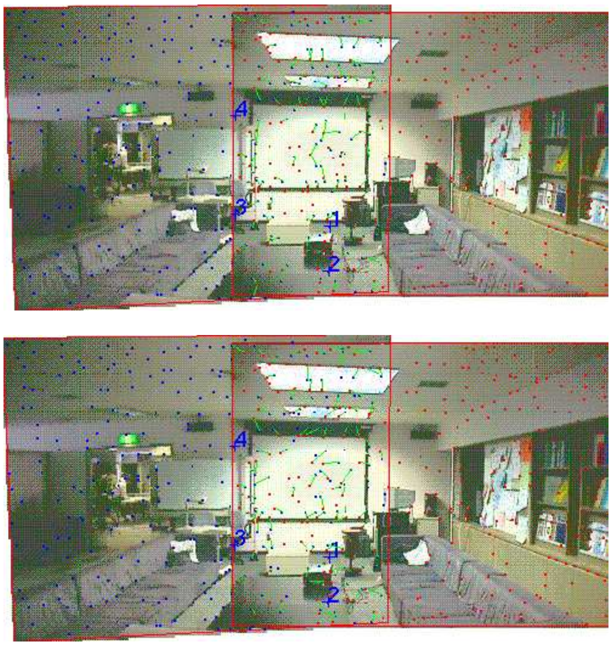

Fig. 3 Haussdorf matching (top). Bottlebeck matching (down).

(4) Discrete approximate matching. Since our point sets are dense and of bounded diameter (namely the image diameter), we can use bucketting techniques. For each point $L \in \mathbf{H}\mathcal{S}_1$ we check whether there is a point $R \in \mathcal{S}_2$ in its neighborhood (see Figure 4).

We report the number of matching points approximately. (Buckets introduce an inherent $\sqrt{2}$-approximation factor). Note that the motif of the boolean bucket can be computed beforehand and does not depend on $\mathbf{H}$ but on $\mathcal{S}_2$.

Running times.
(1) can be computed in time proportional to the number of pixel correlations (window sizes, also proportional

[†] A matching is a set of pairwise disjoint edges.

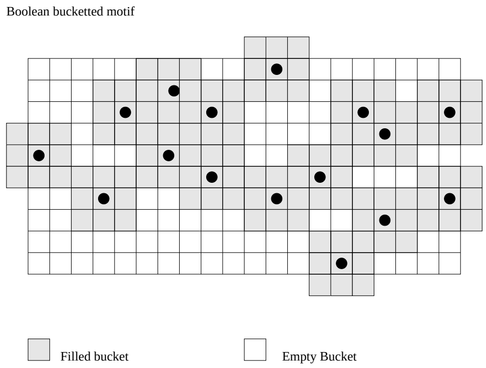

Fig. 4 Boolean bucket on set $\mathcal{S}_2$: grey buckets are marked to contain a point of $\mathcal{S}_2$ in their neighborhood.

to the number of features). (2) can be computed efficiently in $O(n \log n)$-time using Voronoi diagrams but is not appropriate for different scalings. (We may use the hardware graphics pipeline as suggested in [9] to accelerate this computation.) (3) requires more processing time; Efrat and Itai [8] 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. More precisely, 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 [10] 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 $O(m\sqrt{n})$ [11]. When considering geometric graphs, i.e., graphs obtained from a geometric scene, Vaidya [12] 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. [13] to $O(n^{2+\gamma})$ for any arbitrary small positive $\gamma > 0$. Very recently, Efrat and Itai [14] 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 [15]. (See also the work of Heffernan and Schirra [16] for approximation schemes.)

(4) is evaluated in linear time but only gives an approximation of the size of the common point set.

Another classic approach to point set pattern matching, first developed by Huttenlocher et al. [17], is to perform a *branch-and-bound strategy* on some search space; For example, the coefficients of the matrix coding a planar transformation, define a 4-dimensional space (rotation, translation and uniform zoom). The algorithms start with a given $4d$ box containing an optimal solution, splits the current box into sub-boxes, kill some of them (those where the current best solution is better than any solution provided by them) and branch on the remaining active sub-boxes. 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. [18] and, Hagedoorn and Veltkamp [19].

To sum up, when scoring transformation $\mathbf{H}$, we first check that we have a large common point set using (4). If so, we refine the score by using either (2) or (3) depending on the zoom factor of $\mathbf{H}$. Finally, we spend more time computing the score by applying (1) for the remaining transformations.

In the following section, we focus on how to report a pool of candidate transformations in which our solution is likely to be.

4. Geometric filtering

The basic idea of *geometric filtering* is to constrain the properties of feature matchings. We can input geometric constraints like zoom value (e.g., in range $[\frac{1}{4}, 4]$) and rotations (e.g., between $[-60\deg, 45\deg]$), and a precision tolerance ϵ (each feature can be moved into a ball centered at it of radius ϵ). Loosely speaking, we are interested in finding *large common point sets*, that is a set of transformations that match at least a number of points greater than a given threshold. (For example if we want to recover the epipolar geometry, we may ask for at least 7 matching pairs of points). We do not choose a largest common point set because of matching artefacts† but it is very likely that our transformation lies in the ones matching large point sets.

If no value of the threshold of the size of a large common point set is given, the system can estimate it in order to run into given time bounds. For any planar transformation T, we associate a characteristic vector $v(T) = (zoom(T), angle(T))$, where $zoom(T)$ is the zoom value of T and $angle(T)$ is the rotation angle from the x-axis. The algorithm can report lexicographically the transformations $T_1, T_2, ...$ (that is such that $v(T_1) \leqq v(T_2) \leqq ...$) so that two job processes can run simultaneously: (1) reporting potential transformations (2) scoring the transformations and determining if a plausible transformation has been found so far.

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, *inliers*, and possibly occluding parts (hidden features or *outliers*). For example, a 50% overlap

†We mean that the transformation exhibiting a largest common point set may not necessarily be the one being used for mosaicing the images. However the mosaicing transformation matches large common point sets.

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 ($1 \leqq k \leqq 5$) which defines an (α,ϵ)-match. We used the pseudo-random generator `drand48()`.

at constant zoom with 50% of outliers of $n = 40$ point features, will match around 10 features (*ie.* $\alpha = 0.25$). We are looking for a (α,ϵ)-match, i.e. a transformation $\mathbf{H}$ that matches at least αn points up to some error tolerance ϵ.

Algorithm 1 The core selection algorithm

$\mathbf{H} = Id$
while not found a (α,ϵ)-matching homography $\mathbf{H}$ **do**
  Choose r points $\mathcal{S}_1'$ from $\mathcal{S}_1$
  Draw at random a k-tuple $\mathcal{P}_1$ from $\mathcal{S}_1'$
  while not STOP **do**
    Draw at random a k-tuple $\mathcal{P}_2$: $k = |\mathcal{P}_2|$ points of $\mathcal{S}_2$
    (* We use **geometric FILTERING** *)
    for all permutations $\mathcal{P}_2'$ or $\mathcal{P}_2$ **do**
      Compute the homography $\mathbf{H}$ that perfectly matches tuple $\mathcal{P}_1$ to tuple $\mathcal{P}_2'$
      if $\mathcal{S}_1'$ is a (λ,ϵ)-match in $\mathcal{S}_2$ **then**
        if $\mathcal{S}_1$ is a (α,ϵ)-match in $\mathcal{S}_2$ **then**
          if the characteristics of matched points correlate satisfactorily **then**
            STOP
          end if
        end if
      end if
    end for
  end while
end while

Let k be the number of features required in $\mathcal{S}_1$ and in $\mathcal{S}_2$ for computing a basic transformation that perfectly matches pair by pair these $2k$ features (edges, corners, triple junctions, etc.). Using corners, we have k set to 4. Since we know that a significant proportion 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. Table 1 shows experimentally the number of times we loop before finding a good k-tuple (independent of n using the pseudo-random generator `drand48()`).

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 significantly. 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 have approached by p ($p^* \in \mathcal{B}(p,\mu)$). Given any two feature points, we have $|d_2(p_1^*,p_2^*) - d_2(p_1,p_2)| \leqq 2\mu$. Let $L_1, L_2 \in \mathcal{S}_1$ be corner points in image $\mathbf{I}_1$ (resp. image $\mathbf{I}_2$) that have been drawn randomly. Assuming uniform scaling factor, instead of comparing (L_1, L_2) to all pairs (R_i, R_j), we choose for *every point* $R_i \in \mathcal{S}_2$ as candidate for the second point R_j, all the points inside the ring whose center is R_i with minimum circle $\mathcal{B}(R_i, d_2(L_1,L_2) - 2\mu)$ and width 4μ (see Figure 5). This can be done easily using buckets (as depicted in Figure 5) and extend naturally to zoom ranges and angle sectors. We balance the preprocessing time of building the buckets with the processing time of querying it according to the intrinsic difficulty of the point set (see [1] for details). The idea depicted in Figure 8 is that we can count possibly faster than reporting points inside the annuli query. Therefore we can adapt the size of the buckets in order to accomodate the batched ring queries faster. Indeed, loosely speaking, having a too fine bucket costs much preprocesing time but answer queries quickly, while having a coarse-sized bucket is fast to build but queries take more time as illustrated in Fig 8.

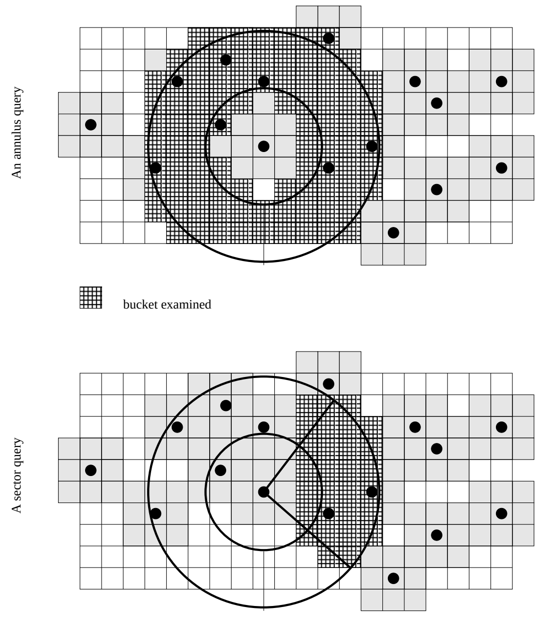

Fig. 5 Ring and sector queries on a bucket. A query behaves as a graphics filling procedure, starting from an initial bucket contained in the geometric ring and examining step by step all the neighbors inside the geometric query until none are left.

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. Erdös conjectured that the $\Omega(n^{1+\frac{c}{\log\log n}})$ bound is tight. This conjecture is related to the self similarity of a point set and arithmetic properties.

We notice that we avoid testing most of the pairs (for example with $n_1 \cong n_2 = 1000$ 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 and feature types. We may also increase k so that for example we require at least 5 matching points. (We compute the homography that minimizes the pairing error.) Figure 6 illustrates the query for $k = 3$ points for a planar transformation.

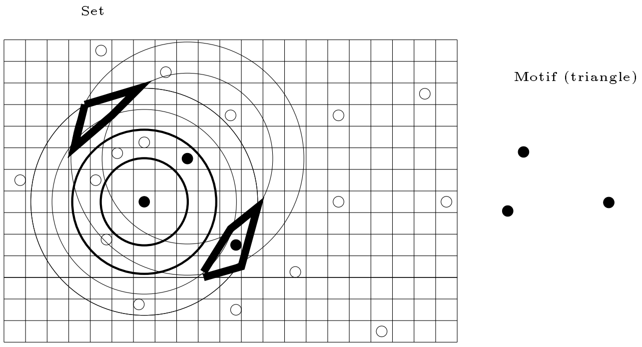

Fig. 6 Looking for at least 3 matching points by querying. The bold regions indicates possible locations of the third point R_3 in $\mathcal{S}_2$ given already two selected features R_1, R_2.

The other natural way to randomize, as already noticed by Irani and Raghavan [20] (for alignment and geometric hashing problems), is to avoid testing the entire set $\mathcal{S}_2$ but rather 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 test whether λr points of $\mathcal{S}'_1$ match $\mathcal{S}_2$ (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 [20] of a point set. Putting it all together, we get the probability of failure to report a transformation when it exists after the ith iteration bounded as follows:

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

As a corollary, for any $\alpha > 0$, we can appropriately choose λ, r and i such that the algorithm will fail with

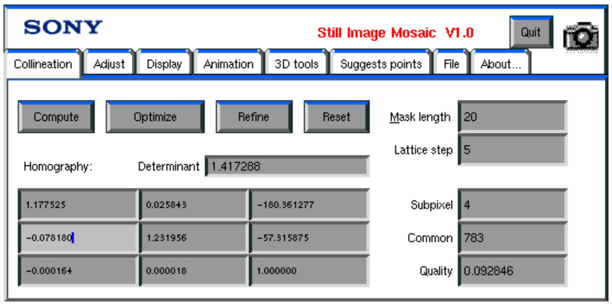

Fig. 7 User interface for the XSTILL system.

probability at most $\frac{1}{2^q}$ by only multiplying the overall cost by $O_{\lambda,\alpha,r,k}(q)$. We use this system with $k = 2$ (and ϵ around 6 pixels). Once a rough planar transformation is found, we label inliers/outliers and compute the homography that minimizes the pairing of corresponding inliers (using for example Kanatani's approach [21]).

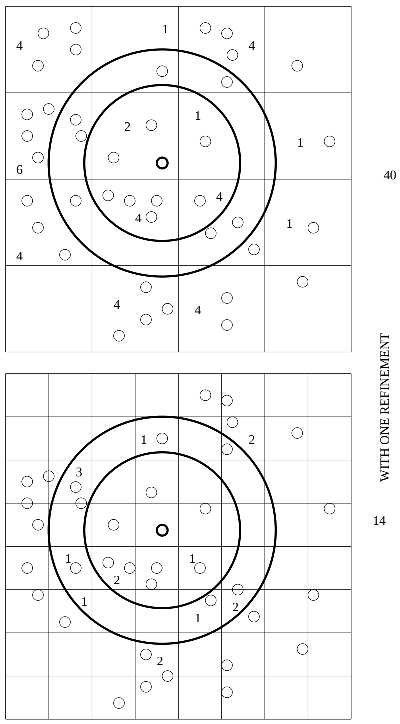

Fig. 8 Adjusting the width of a bucket. (Top) a 4x4 bucket used to report a superset of size 40 points containing the 7 points inside a geometric ring. (Bottom) After a refinement,a 8x8 bucket used to report a superset of size 14 points containing the 7 points inside a geometric ring.

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 [22]. 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], [15] 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.

5. Deghosting

In practice, it is difficult to take still digital snapshots while maintaining the position of the optical center fixed without tripod (*id.*, only rotating around the optical center). Therefore the mapping of $\mathbf{I}_1$ onto $\mathbf{I}_2$ is only approximated by a noisy homography (parallax errors). One way to cope with this problem is to compute a Delaunay triangulation of the features that match $\mathcal{S}_1$ and $\mathcal{S}_2$ (inliers). (If segments are part of the feature sets, we may ask for a constrained Delaunay triangulation.) Also, we can compute the epipolar geometry and, once the fundamental matrix coding the 3d translation, rotation and scales between the two images is known, each triangle of the triangulation of $\mathbf{I}_1$ maps onto a corresponding triangle of $\mathbf{I}_2$. The induced transformation between $\mathbf{I}_1$ to $\mathbf{I}_2$ is then computed piecewise. However, we have not implemented this scheme since our purpose was to focus on pattern matching for image registration problem.

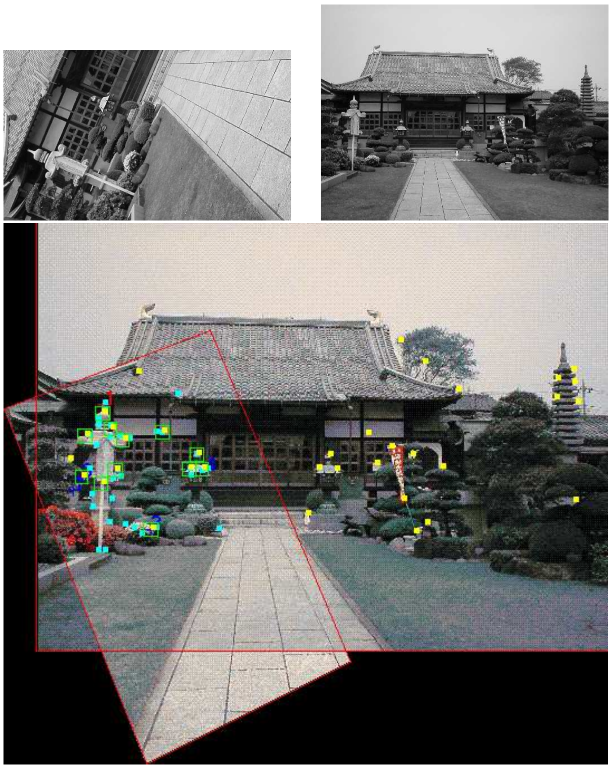

Fig. 9 Mosaicing of two images (zoom=2, rotation=67 deg) (with $\epsilon = 6$ pixels and $n = n_1 = n_2 = 35$).

6. Performance of the system

The system **Xstill** has been implemented on Linux OS using gnu C++ compiler and is about 10K lines of code. The program handles bundles of images as well and the user interface allows the user to possibly interact/set appropriate parameters (see Figure 7). We describe below experimental data. We use `drand48()` as a pseudo-random number generator (vs. true random generator for theoretical analysis). The reported values are subjective since it does depend on the image data set. (Indeed since our method is based on pattern matching, it is nontrivial to establish the pertinence of the extracted feature points with the information contained in the image!)

On a pentium II MMX, with $n_1 = n_2 = 50$, $\epsilon = 2\%$ of image lengths and $\alpha = 50\%$, at constant zoom, the deterministic algorithm takes 1.77sec and test 0.83% of the possible planar transformations. The randomized algorithm takes 0.01sec and test 0.0075% of the possible transformations. At zoom factor 2, the deterministic algorithm selected 3.8% of the transformations in 7.8sec while the randomized algorithm examined 0.07% in 0.17sec. At zoom 4, in 9.8sec, we looked at 4.75% of the possible transformations, compared to a bare 0.3sec for the randomized algorithm that looked at 0.14% of the possible transformations. For larger zoom values, the percentage of inliers tend to decrease quickly because feature extractors become less reliable. Also, it is an open problem to find a 'worse' configuration of the features (related to Erdös' conjecture). Once the homography has been found the overhead for warping and blending images is typically around 1 or 2 seconds.

7. Ongoing research

For large zoom values (greater than 4), the detected features are not that precise (parameter ϵ) and the induced homographies are noticeably unstable. Moreover, the greater the zoom value, the larger the point set size n, and the slower the system. In practice, we would also like to detect parts that have been static (ie., the background) from the ones which have dynamically changed (eg., pictures taken with people walking in an exhibition hall, etc.).

Our paradigm can be used as well for tackling various other problems (like merging 3d depth data sets obtained from range finding or active vision). Some fundamental problems linked to the *large common point set* solver are symmetry, congruency and similarity of point sets. Any improvement of those core algorithms may deliver faster guaranteed pattern matching algorithms, and provide a wider use of feature-based mosaicing.

Acknowledgement

I would like to express my gratitude to Imad Zoghlami [6] (INRIA Robotvis – France) who introduced me to mosaicing systems and to Pr. Mario Tokoro (Sony CSL Inc. – Japan) for supporting my research projects. Finally, I thank the anonymous referees for their insightful comments.

References

[1] F. Nielsen. *Randomized Adaptive Algorithms for Mosaicing Systems*, SCSL-TR-98-0035, Technical Report, Sony Computer Science Labs. `http://www.csl.sony.co.jp`, 1998.

[2] A. Goshtasby, R. Szeleski, and G. Wolberg. *2D and 3D Image Registration and Image Warping*, chapter Course note 2. Siggraph, 1999.

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

[4] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic mosaics and environment maps. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 251–258. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[5] Steve Mann and Rosalind W. Picard. Video orbits of the projective group: a simple approach to featureless estimation of parameters. Technical Report 338, MIT Media Lab., 1996.

[6] I. Zoghliami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2D mosaic from a set of images. In *CVPR Proceedings*, 1997.

[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] Kenneth E Hoff, III, Tim Culver, John Keyser, Ming Lin, and Dinesh Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. Technical Report TR99-011, Department of Computer Science, University of North Carolina - Chapel Hill, February 15 1999. Mon, 1 Mar 1999 20:34:34 GMT.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15:850–863, 1993.

[18] 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.

[19] 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.

[20] 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.

[21] Kenichi Kanatani. Optimal Homography Computation with a Reliability Measure. In *MVA'98 IAPR Workshop on Machine Vision Applications*, pages 426–429, 1998.

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

Frank Nielsen was born in France in 1971. He received the B.S. and M.S. degrees from École Normale Supérieure (ENS) of Lyon in 1992 and 1994, respectively. He defended his Ph. D. thesis on "Adaptive Computational Geometry" prepared at INRIA Sophia-Antipolis under the supervision of Pr. Boissonnat in 1996. As a civil servant of the University of Nice (France), he gave lectures at the engineering schools ESSI and ISIA (École des Mines). In 1997, he served army as a scientific member in the computer science laboratory of École Polytechnique (LIX). In 1998, he joined Sony Computer Science Laboratories, Tokyo (Japan) as an associate researcher. His current research interests include computational geometry, algorithmic vision, combinatorial optimization for geometric scenes and compression. His e-mail address is `nielsen@csl.sony.co.jp`