Robust Random Dot Markers: Towards Augmented Unprepared Maps with Pure Geographic Features

Liming Yang* Jean-Marie Normand† Guillaume Moreau‡

CERMA UMR CNRS 1563, École Centrale de Nantes, Nantes, France

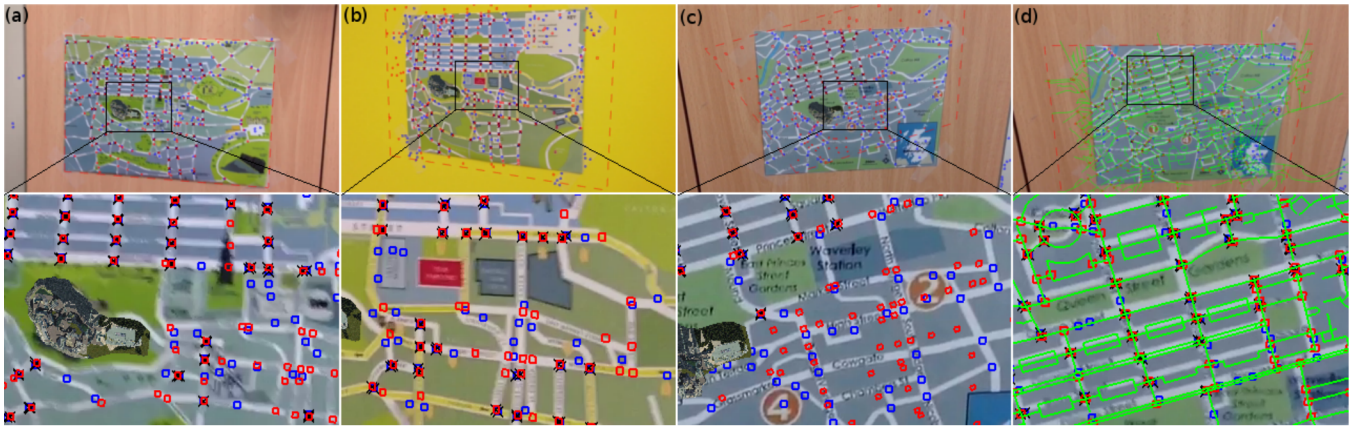


Figure 1: *Augmenting natural maps of Edinburgh. Maps (a), (b) and (c) are three different maps registered with a single database and a 3D model of the castle is rendered onto them. Map (d) is registered with real GIS data and a GIS road network (green) is reprojected onto it. Blue points represent detected intersections, red points represent reprojected ground truth and black crosses represent inliers.*

Abstract

Augmented maps have many important applications. However, no mature registration method exists to associate unprepared maps with a Geographical Information System (GIS) database which would be used to superimpose simulation results or route display on a paper map. In this paper, we propose a method called Robust Random Dot Markers (RRDM) that can robustly track coplanar random dot patterns, which can be used to address this problem. RRDM is based on the same idea of the Random Dot Markers (RDM) proposed by [Uchiyama and Saito 2011] and it can serve as fiducial markers as well as texture independent "natural markers". We conduct a series of experiments and show that RRDM is more robust than RDM in terms of jitter, perspective distortion, under and over detection of dots in the pattern. As an example of "natural marker", we show that RRDM can successfully register unprepared printed maps only with pure geographic features, i.e. road intersections coordinates, which we retrieve from a GIS. Our method does not suffer from the drawbacks of traditional "feature-point" based registration methods which mainly based on textures, since textures may change according to different maps.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking;

*e-mail: liming.yang@ec-nantes.fr
†e-mail: jean-marie.normand@ec-nantes.fr
‡e-mail: guillaume.moreau@ec-nantes.fr

Keywords: tracking, point pattern matching, augmented maps

1 Introduction

Printed maps are an essential and widely available media to present geographic information, especially in urban areas. They are used for several tasks ranging from route finding to local authorities decision making. Although electronic maps have emerged nearly a decade ago, paper maps are still widely used thanks to their unrivaled advantages. Users can have natural interactions with paper maps such as adding annotations or drawing trajectories. It is also easier for several users to collaborate on a common paper map which facilitates communication. However, unlike electronic maps, paper maps remain static: their contents cannot be changed according to users' preferences, no computation result (such as flow simulation or shortest path) can be added without adding layers or reprinting the map. We believe that augmented maps, on which dynamic digital information is added on top of traditional paper maps, may combine both advantages of paper maps and electronic maps.

To build augmented maps, we need to track different unprepared printed maps of the same city with respect to existing databases, such as Geographic Information Systems (GIS) which will provide data for further augmentation. The most difficult task is to georeference the target map during initialization, in other words, to find the relationship between positions on the map and their geographic coordinates. After the initialization is done, traditional tracking methods can be easily implemented.

VRST 2014, November 11 – 13, 2014, Edinburgh, Scotland, UK.

http://dx.doi.org/10.1145/2671015.2671022

Since maps are rich textured planar objects, one can track them using "natural features" based on local textures, such as SIFT [Lowe 2004]. However, this method is texture-dependent and thus map-dependent while GIS mainly contain vector data (lines, points, polygons, etc.). Traditional matching methods based on local textures are thus useless for geo-referencing with GIS. We can only rely on pure geometric features, e.g. road networks.

To address this limitation, we developed a new method called Robust Random Dot Markers (RRDM), which identifies and localizes coplanar point patterns (i.e. markers) under large camera perspective distortion. Markers' discrimination is achieved by the randomness of the distribution of points as it has been proved by a document recognition application in [Nakai et al. 2006]. Our method is robust against inaccuracy of point positions (*jitter*), partial occlusions, as well as presence of extra and missing points in the marker. When the number of points is moderate, real-time tracking can be achieved. Tracking by point pattern matching is also a general approach to track natural scenes that contain randomly distributed points (such as corners). [Uchiyama and Marchand 2011] proved it by augmenting various planar objects (with and without texture information, handwriting, ARToolkit markers, etc.). As our method significantly improves the robustness of point matching, it can be used in more complex situations. We show in this paper that natural map tracking based on real GIS data is a good candidate application.

We reformulate the point pattern matching problem mathematically as follows: let P_r be a reference point set in 2D coordinates; H an unknown 2D homography. Let P_i be a set of *observed* image points, of which some points belong to $H(P_r)$ but with small differences in their positions, while others are randomly added noise points (i.e. extra points). The problem is to determine H given P_r and P_i. In our map case, P_r refers to an existing database of road intersections and P_i represents detected road intersections. We assume that points in P_r are uniformly randomly distributed, as is the case in most European countries. The challenges of the problem are large point jitter, incomplete matching and the eight degrees of freedom induced by a perspective transformation.

The remainder of this article is structured as follows: Section 2 is dedicated to related works while Section 3 describes our TSR descriptor. The local voting and coherency mechanism is detailed in Section 4. Section 5 discusses the algorithm's parameters. Experimental results and comparisons are presented in Section 6. Finally Section 7 draws some conclusions and discusses future work.

2 Related works

2.1 Planar object tracking

Fiducial markers are man-made objects, often planar, placed in the field of view to serve as reference points. The two most representative fiducial markers are square and circular markers. Square markers use four corners of a rectangle as points of interest to estimate perspective transformation. As a single marker is sufficient to compute the camera pose, they are also denoted as "planar" markers [Lepetit and Fua 2005]. Alphabet letters [Kato and Billinghurst 1999] and 2D bar codes [Rekimoto 1998] are also used as marker identifiers. Regarding circular fiducial markers, at least four of them positioned in a coplanar but non collinear way are required to compute a unique perspective transformation, thus they are referred to as "point" markers [Lepetit and Fua 2005]. Contrasting concentric circles (CCC) [Gatrell et al. 1992] is the main trend of this type of markers for which sub-pixel accuracy can be achieved. [Uchiyama and Saito 2011] recently proposed Random Dot Markers (RDM). They use local arrangements of points to construct local descriptors for matching, known as LLAH [Nakai et al. 2006]. Their method proves to be robust to partial occlusions.

Besides fiducial markers, it is common to use natural features, such as SIFT [Lowe 2004], SURF [Bay et al. 2008] or BRIEF [Calonder et al. 2012]. Such methods firstly look for interest points in both the template and the target object. Then, for each point of interest, a high dimension descriptor is constructed using a local image patch. Correspondences can then be established by matching these descriptors and thus the target object can be tracked.

2.2 Map tracking

Paper map tracking with fiducial markers demands special preparations on maps beforehand. [Reilly et al. 2006] used flat RFID tags attached at landmark positions behind a map to identify different locations on the map. [Schöning et al. 2006] inserted ARToolkitPlus [Wagner and Schmalstieg 2007] markers on the map that contain its geodetic coordinates and orientation. Using the RDM approach, streets intersections [Uchiyama et al. 2011; Uchiyama et al. 2009] or cities landmarks [Martedi and Saito 2011] are manually marked and used as random dot patterns.

Natural features have also been used: [Morrison et al. 2011] used a modified SIFT descriptor to achieve feature points matching while [Reitmayr et al. 2005] used a gradient histogram similar to SIFT. None of them can provide an appropriate solution in the absence of relevant texture information.

[Jiang et al. 2011] proposed to use road intersections for geo-referencing a markerless map. They first detect road intersections from target maps before registering them by geometric hashing with reference ones extracted from a GIS. Nevertheless their method cannot take into account warping due to perspective.

2.3 Point pattern matching

Incomplete point pattern matching with the presence of point jitter is also a subject of interest in the pattern matching community. Traditional methods, such as Generalized Hough Transformation [Ballard 1981], pose clustering [Olson 1997] and Geometric Hashing [Wolfson and Rigoutsos 1997] have huge computational complexities when the degree of transformation is high. [Jiang et al. 2011]'s approach is not suitable for our problem because it only takes into account a similarity type transformation. RANSAC [Fischler and Bolles 1981] is another frequently used method to match two point sets under perspective transformation. It requires at least 50% of inliers to work, which is not our case when dealing with automatic road detections on unprepared maps. PROSAC [Chum and Matas 2005] demands lower inlier ratio, but needs an *a priori* quality for each correspondence describing how likely it is for a correspondence to be an inlier. Many recent papers focus on rigid transformations and related works can be found in [Van Wamelen et al. 2004]. [Aiger and Kedem 2010] proposed a new algorithm applicable to any group of transformations which, however, requires the points forming the chosen basis to be always present in the scene. [Datta et al. 2013] recently proposed a robust method using a local geometric invariant, but they mainly deal with affine transformations. To the best of our knowledge, this work presents the first point pattern tracking algorithm suitable to the presence of point jitter, extra and/or missing points in the pattern area under large perspective distortions.

3 Quad-Point and its descriptor

We know that (a) pure point pattern matching can only rely on the geometrical pattern itself; and that (b) presence of extra/missing

points and point jitter can significantly alter local geometry of the point pattern. Our task is difficult since we have to take into account both (a) and (b) at the same time. A RDM descriptor is an integer composed of five surface ratios computed from seven neighboring points. Any change in those seven neighbors can modify the descriptor, making it very sensitive to noise. We use as few points as possible to build the descriptor, so that it is less likely to be influenced by (b). We choose four points (a so-called "Quad-Point") since fewer points cannot provide any affine invariant. In this section, we define the "TSR" descriptor and show how two "jittered" Quad-Points can be matched under an affine transformation which lays the groundwork for our new algorithm.

3.1 Definition

We call any four-coplanar-point configuration a "Quad-Point". We do not consider the situation where three points are aligned since it rarely occurs for randomly distributed points. Each Quad-Point is composed of a main point M and three associated points F, R, L. Under such definition, every Quad-Point can be categorized into one of three possible configurations c. For each configuration, the spatial relationships between these four points, namely M (main), F (face), R (right), L (left), are illustrated in Fig. 2. Especially, when $c=3$, F is chosen so that the surface of ΔMRL is the second smallest among the four triangles. As a consequence, a Quad-Point, denoted as $Q =< M;F,R,L >$ (note that the order of the points is important), is uniquely defined when four coplanar points are given and its main point M chosen (we therefore say that the Quad-Point belongs to M and denote it as $Q(M)$ when all four points are known).

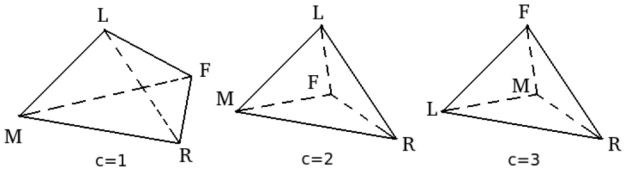


Figure 2: *Quad-Point configurations.*

Two Quad-Points $Q =< p_1;p_2,p_3,p_4 >$ and $Q' =< p'_1;p'_2,p'_3,p'_4 >$ are affinely equivalent if there exists an affine transformation $\mathscr{A}$ between them.

$$Q \equiv Q' \Leftrightarrow \exists \mathscr{A} \text{ s.t. } p_s = \mathscr{A}(p'_s), s = 1,2,3,4 \tag{1}$$

All Affinely Equivalent Quad-Points (AEQ) can be grouped together. Since a Quad-Point contains exactly two independent invariants [Hartley and Zisserman 2004], each AEQ group has two degrees of freedom.

We propose to rely on two surface ratios along with a configuration specifier to describe each AEQ group. Given its nature, we name our descriptor Two-Surface-Ratio (TSR). In this context, $Q(M)$'s TSR descriptor, $\mathbf{t}(Q(M))$, is defined as follows:

$$\mathbf{t}(Q(M)) = (X,Y,c) \tag{2}$$

$$with \begin{cases} X = \dfrac{S_{\Delta FRL}}{S_a}, Y = \dfrac{S_{\Delta MFL}}{S_a}, & \text{for } c = 1,2 \quad (3) \\ X = \dfrac{S_{\Delta MRL}}{S_a}, Y = \dfrac{S_{\Delta MFL}}{S_a}, & \text{for } c = 3 \quad (4) \end{cases}$$

where S_a is the surface formed by the convex hull of Q.

We prove in Section 8 that two Quad-Points are affinely equivalent if and only if their TSR descriptors are equal. Note that in absence of point jitter, when two Quad-Points are matched under an affine transformation, they have the same TSR descriptor.

3.2 Point jitter and descriptor variance

Point jitter can be imputable to two possible sources : (i) imaging system's distortions and (ii) misdetection of points in the image. The former case can be induced by lens distortions and is both scene and scale independent. In the latter case, point jitter is scale dependent, becoming smaller when the scene gets further away from the camera. For narrative convenience, in the following we only consider point jitter as arising from the second source.

With the presence of point jitter, the form of Quad-Points can be changed. The original Quad-Point and its *jittered* version are generally not affine equivalent, thus they can not be exactly matched by (1). However, when jitter is not too significant, these two Quad-Points should be "quasi-affine-equivalent". We show here that this quasi-affine-equivalence can be measured by the TSR descriptor.

We assume that point jitter can be modeled as an additive Gaussian noise to the points' original positions before perspective transformation. Mathematically, each point in the Quad-Point is affected by an additive Gaussian noise $\delta \sim N(0,\sigma^2)$ in u and v directions independently (cf. Fig. 3), where σ is small compared to distances between points forming the Quad-Point.

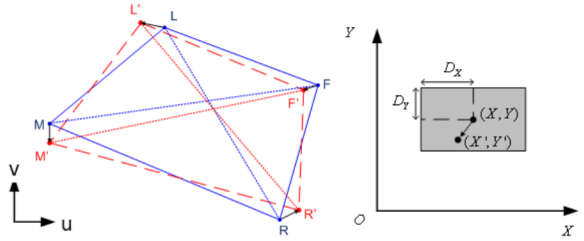


Figure 3: *Jittered Quad-Point and its TSR descriptor. Left : Original Quad-Point Q (solid blue line) and jittered Quad-Point Q' (red dotted line) in a front view. Right : Original (X,Y,c) and jittered (X',Y',c) descriptors in descriptor space. Gray region represents the matching region $R(X,Y,c)$.*

Under such assumptions, we can compute the descriptor's variance in first order approximation. Fig. 3 illustrates an example of original and *jittered* Quad-Points as well as the related descriptors. For illustration purposes, we compute d_X, the difference between the original Quad-Point's TSR descriptor component X and the jittered one's X', when $c=1$:

$$\begin{aligned} d_X =& X' - X = \frac{S_F dS_M - S_M dS_F}{(S_M + S_F)^2} \sim N(0,\Sigma_X^2) \\ \Sigma_X^2 =& \frac{\sigma^2}{4(S_M+S_F)^4}\{S_M^2\left(\overline{FR}^2 + \overline{FL}^2 + \overline{RL}^2\right) \\ & + S_F^2\left(\overline{MR}^2 + \overline{ML}^2 + \overline{RL}^2\right) \\ & - S_M S_F\left(\overline{MR}^2 + \overline{ML}^2 + \overline{FR}^2 + \overline{FL}^2 - 2\overline{MF}^2\right)\} \end{aligned} \tag{5}$$

where $S_M = S_{\Delta FRL}$ and $S_F = S_{\Delta MRL}$.

So, when a Quad-Point is slightly perturbed by δ with a small σ, its TSR descriptor will mostly remained unchanged. Thus, we can define two Quad-Points Q and Q' such that they are quasi-affinely-equivalent ($Q \simeq Q'$) thanks to our TSR descriptor.

$$Q \simeq Q' \Leftrightarrow |X - X'| \leq D_X, |Y - Y'| \leq D_Y \text{ and } c = c' \tag{6}$$

with $t(Q) = (X, Y, c)$ and $t(Q') = (X', Y', c')$. Practically, we set D_X (resp. D_Y) as follows :

$$D_X = \begin{cases} 2\Sigma_X, & \text{if } 2\Sigma_X \leq D_{max} \quad (7) \\ D_{max}, & \text{otherwise} \quad (8) \end{cases}$$

For clarity sake, we say two Quad-Points are quasi-affinely-equivalent when the TSR descriptor of the one is inside the matching region of the other. Matching region of Q is represented by a gray box on the right of Fig. 3. In the following, we consider that two Quad-Points are *matched* if they satisfy (6).

4 Algorithm

Throughout the paper, we use the following conventions: P_r is the reference point set, it contains the coordinates of m reference points and corresponds to the point pattern that is looked for in the image. P_i is the image point set which contains the coordinates of n image points that are detected in the image. Each pair (p_r, p_i), with $p_r \in P_r, p_i \in P_i$ is a tentative correspondence (or in short correspondence) that could either be an inlier (i.e. a true correspondence) or an outlier (i.e. a false correspondence).

Both point jitter σ and point pattern density $\bar{\rho}$ (i.e. the number of points per unit area) play an important role in Quad-Point matching. Given a fixed jitter value σ, the denser the point pattern, the bigger descriptor variance, cf. (5). We define the *jitter factor* $\eta = \sigma\sqrt{\bar{\rho}}$ to describe the amount of point jitter (cf. Section 3.2).

The main idea of our approach is that even under a large perspective distortion, the transformation between P_r and P_i can be approximated by an affine transformation in a small region. We use local affine information and consistency between neighboring regions to establish point correspondences before estimating an homography.

Our method can be decomposed into two stages. The first one is in charge of registering every reference point into hash tables. This stage is performed once offline before online tracking. During the second stage, we retrieve P_i from each image, and match it with P_r by using a local voting and coherency (LVC) strategy.

4.1 Offline pre-registration

The offline pre-registration stage is detailed in Algorithm 1. P_r is loaded from a database representing the point pattern. For each point $p_r \in P_r$, we find its k-nearest neighbors $N_k(p_r)$. Each point p_r is associated with a set of Quad-Points $QS(p_r)$, defined as follows:

$$QS(p_r) = \{Q(p_r) =< p_r; p_1, p_2, p_3 >, \forall p_1, p_2, p_3 \in N_k(p_r)\} \quad (9)$$

The order of p_1, p_2, p_3 is predefined, so each set of Quad-Points $QS(p_r)$ contains $C_k^3 = \frac{k!}{3!(k-3)!}$ Quad-Points.

We use three bi-dimensional $B \times B$ hash tables to register reference points so that they can be quickly matched during the online stage with image points according to their descriptors. Each hash table corresponds to one configuration of Quad-Points illustrated in Fig. 2. We determine the set of hash bins $S_{idx} = \{(i, j, c)\}$ into which a Quad-Point Q_r with TSR descriptor $\mathbf{t}(Q_r) = (X, Y, c)$ is registered as follows, with $[\bullet]$ represents a floor function:

$$\begin{aligned} [(X - D_X)B] \leq i \leq [(X + D_X)B] + 1, i \in N \\ [(Y - D_Y)B] \leq j \leq [(Y + D_Y)B] + 1, j \in N \end{aligned} \quad (10)$$

For clarity sake, imagine that the square $[0,1] \times [0,1]$ is uniformly divided into B^2 subregions by vertical and horizontal lines as illustrated in Fig. 4. Each subregion represents a hash bin. We register the Quad-Point with $\mathbf{t} = (X, Y, c)$ into the hash bins which intersect the matching region $R(X, Y, c)$.

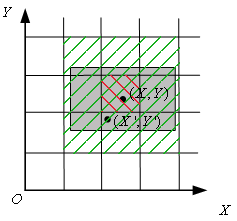


Figure 4: *Quantization of descriptor value for configuration c. The subregion with red stripes represents $H_{idx}(X, Y, c)$ while subregions with green stripes represent $S_{idx}(X, Y, c)$.*

Algorithm 1 Offline Reference points pre-registration

```
for p_r ∈ P_r do
    Extract QS(p_r) according to (9)
    for Q_r ∈ QS(p_r) do
        Calculate t(Q_r) from (2)
        Register Q_r at all hash bins in S_idx according to (10)
    end for
end for
```

4.2 Local Voting and Coherency

This stage can be decomposed into three phases: (i) pre-alignment by local voting, (ii) rejection by local coherency and (iii) recovery. During the local voting phase, we establish the pre-alignment by computing a score for each correspondence between P_r and P_i. During the rejection phase, we calculate a local affinity for each correspondence using their neighboring points. We discard correspondences for which affinities are not consistent with those of their neighbors. During the recovery step, we try to find inliers that were potentially wrongly rejected in phase (ii).

4.2.1 Local Voting

Similar to Section 4.1, a set of Quad-Points $QS(p_i)$ is generated for each point $p_i \in P_i$. The computational cost is $O(kn \log n + k^3 n)$.

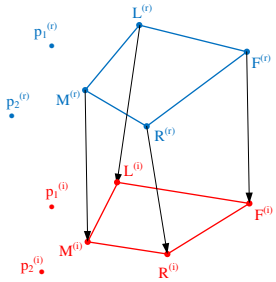
Then for each Quad-Point Q_i, a set of reference Quad-Points QM is retrieved from hash bin H_{idx} according to its TSR descriptor (X, Y, c). These matching results are recorded into voting tables. H_{idx} is calculated as follow, with $[\bullet]$ represents a floor function:

$$H_{idx} = ([BX], [BY], c) \quad (11)$$

From (6), (10) and (11), we are sure that all reference Quad-Points which can be matched with Q_i are contained in QM.

For each tentative correspondence M_r and M_i, we use a $k \times k$ matrix to record their k-neighbor's corresponding relations, denoted by T_{M_r, M_i}. These relations are established from two matched Quad-Points presented in Fig. 5. The detailed process is described in Algorithm 2. As each matrix only represents the correspondences between two subsets, we call it local voting.

Since each image Quad-Point Q_i has to be tested against each Quad-Point in QM, the time complexity is $O(k^3 nr)$, where the redundancy number r is the average size of QM across all hash entries.



M(r),M(i)	L(i)	F(i)	p1(i)	R(i)	p2(i)
p1(r)	3		4		
L(r)	5+1	4	2	1	
p2(r)			2		3
F(r)		4+1		3	1
R(r)	1			4+1	

Figure 5: *An increment of vote for table of T_{M_r,M_i} from two matched Quad-Points. Left : Neighbor correspondences are established from two matched Quad-Points. Right : Corresponding cells are incremented by one vote in T_{M_r,M_i}.*

After local voting, the "quality" (cf. [Chum and Matas 2005]) of each tentative correspondence can be represented by the sum of the votes in its voting table. With this measure, a PROSAC-like algorithm can be implemented. However, this approach does not suit our problem as shown in the experiments of Section 6.

Algorithm 2 Local voting procedure

for $p_i \in P_i$ **do**
  Extract $QS(p_i)$ according to (9)
  for $Q_i =< M_i; F_i, R_i, L_i > \in QS(p_i)$ **do**
    Calculate $\mathbf{t}(Q_i)$ from (2)
    Calculate $H_{idx}(Q_i)$
    Retrieve QM at H_{idx} from hash table
    for all $Q_r =< M_r; F_r, R_r, L_r > \in QM$ **do**
      $T_{M_r,M_i}(F_r, F_i)$++
      $T_{M_r,M_i}(R_r, R_i)$++
      $T_{M_r,M_i}(L_r, L_i)$++
    end for
  end for
end for

4.2.2 Rejection

This phase aims at rejecting outliers and is detailed in Algorithm 3. First we clear votes that are lower or equal to a c_{vote} threshold (which value is discussed in Section 5.2):

$$T_{p_r,p_i}(\tilde{p}_r, \tilde{p}_i) = 0, \text{ if } T_{p_r,p_i}(\tilde{p}_r, \tilde{p}_i) \leq c_{vote} \tag{12}$$

We then estimate an affinity for each tentative correspondence and group those that have similar affinities into an *affinity group*. Using a SVD decomposition, we extract two scaling factors d_{u^*}, d_{v^*} along two principal axes and one rigid rotation θ from any affinity $\mathscr{A}$. We consider two affinities to be similar ($\mathscr{A}_1 \approx \mathscr{A}_2$) if they satisfy:

$$\begin{aligned} &\Delta\theta \leq \theta_c, \text{with } \Delta\theta = |\theta_1 - \theta_2| \\ &\alpha_c^{-1} \leq \alpha_u, \alpha_v \leq \alpha_c, \text{with } \alpha_u = |\frac{d_{u_1^*}}{d_{u_2^*}}|, \alpha_v = |\frac{d_{v_1^*}}{d_{v_2^*}}| \end{aligned} \tag{13}$$

Note that based on (13), θ_c and α_c are two parameters required by our algorithm and are discussed in Section 5.3.

We use the Hungarian algorithm for the maximum assignment problem during Step 1 of Algorithm 3. The Hungarian algorithm has a worst-case time complexity of $O(k^3)$. Nevertheless, it should be noted that since there is a large difference in the number of votes between inliers and outliers in T_{tc} (cf. Section 5.2), the algorithm is able to find the optimal solution in a few iterations. Step 1 of Algorithm 3 is the most expensive part of the algorithm with a worst time complexity of $O(mnk^3)$.

Algorithm 3 Rejection

Step 1. Calculate Affinity
for all $tc = (p_r, p_i) \in$ {All tentative correspondences} **do**
  s = Solution of maximum assignment problem for T_{tc}
  $s_{tc} = s - \{(\tilde{p}_r, \tilde{p}_i)\}$, $\forall T_{tc}(\tilde{p}_r, \tilde{p}_i) = 0$
  if $size(s_{tc}) \geq 4$ **then**
    Estimate $\mathscr{A}_{tc}$ using s_{tc}
  else
    Rejection: mark tc as outlier
  end if
end for

Step 2. Create Affinity Groups (AG)
Group tentative correspondences having similar affinity (13) into the same affinity group g, with $g \in AG$.

Step 3. Estimate homography for each Affinity Group (AG)
for all $g \in AG$ and $size(g) > 4$ **do**
  Estimate H_g using RANSAC
  Discard $(p_r, p_i) \in g$, if $|p_r - H_t^{-1}(p_i)| > 2\sigma$
end for

After this procedure, every correspondence in the same affinity group g can be described by the same homography H_g. Since outliers are randomly matched, it proves almost impossible to estimate the same homography for several outliers. Therefore, we consider that the largest affinity group is the one containing inliers and that it is the first "solution list" (denoted as sl in Algorithm 4).

4.2.3 Recovery

We refine the solution list by gradually adding false outliers rejected during Step 2. This increases the number of inliers and results in a better estimation of the homography between P_r and P_i. The detailed procedure of the recovery phase is shown in Algorithm 4.

Algorithm 4 Recovery

sl = largest $g \in AG$
repeat
  Estimate homography H using sl with least-square method.
  for all $tc = (p_r, p_i) \in sl$ **do**
    for all $(\tilde{p}_r, \tilde{p}_i) \in s_{tc}$ **do**
      if $|\tilde{p}_r - H^{-1}(\tilde{p}_i)| \leq 2\sigma$ **then**
        $sl = sl \cup \{(\tilde{p}_r, \tilde{p}_i)\}$
      end if
    end for
  end for
until No new element added to sl
Return H

5 Choice of parameters

The algorithm presented in Section 4 relies on several parameters. In this Section, we detail the meaning of each parameter and propose some default values so that one can easily adapt our algorithm to one's own applications and needs.

5.1 Robustness of descriptor and k, η, D_{max}

Random addition or removal of points in the close vicinity of a main point M can deeply modify M's k-nearest neighboring points. We call the points not originally present in the neighborhood of a point *false neighbors*. Such false neighbors result in the creation of new Quad-Points in the image which necessarily lead to a reduced number of correct matchings between reference and image points.

For illustration purposes, let us consider $k = 7$ which leads to the creation of $C_7^3 = 35$ Quad-Points for each point in the pattern. In the presence of 25% additional or missing points (uniformly distributed wrt. the original point pattern) in the image, we can assume that each point of interest M will have 2 false neighbors in 7. Thus the remaining original Quad-Points reduce to $C_5^3 = 10$, meaning that about 10 correct matching are left for each point.

So, robustness with respect to additional or missing points can be improved by increasing k, at the expense of impacting computational efficiency of the algorithm (cf. Section 4.2), since most steps have a computational cost in the order of k^3.

Parameters η and D_{max} are used to manage point jitter. When points in the image are likely to have a high incertitude in their positions, η should be set to a higher value. When incertitude is so big that equation (8) dominates over equation (7) (resp. for D_Y), a larger D_{max} should be used. Following those guidelines, one can augment the chances to match a *jittered* Quad-Point correctly, thus increasing the robustness of the algorithm against large point jitter. However, increasing η and D_{max} results in enlarging the matching region and in a higher value of the redundancy number r (cf. Section 4.2) which in turn impacts the efficiency of the algorithm.

5.2 Threshold c_{vote}

Before applying Algorithm 3, correspondences that have received less than or equal to c_{vote} votes in T_{tc} have been rejected. The idea behind this thresholding mechanism is to keep as many inliers as possible while rejecting potential outliers. A brief study of the effectiveness of this threshold is now presented.

Let $Z_i(v)$ and $Z_o(v)$ be two random variables representing the votes respectively received by an inlier and an outlier in voting tables. We find their probability distribution functions $Z_i \sim \phi_i$, $Z_o \sim \phi_o$ experimentally by analyzing 200 randomly distributed points 1000 times with $k = 7$, $D_{max} = 0.2$ and $\eta = 5\%$

Using these parameters values, an important difference in the number of votes received by inliers and outliers is presented in Fig. 6. $\phi_t(1) \approx 0.03\%$ and $\phi_f(1) \approx 17.3\%$ indicate that 0.03% inliers and 17.3% outliers receive 1 vote. The best choice is $c_{vote} = 8$, since $c_{vote} \geq 9$ will not reject more outliers (in proportion) than inliers.

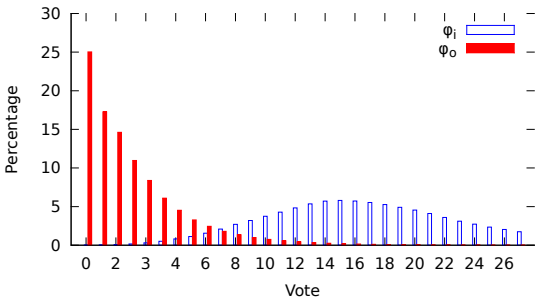


Figure 6: *Vote probability distribution functions (ϕ_i,ϕ_o)*

It is worth noting that the choice of c_{vote} has an impact on both the robustness of the algorithm and its speed. Hence, its value should be chosen according to a trade-off between robustness (lower c_{vote} values) and computational efficiency (higher c_{vote} values).

5.3 Influence of point jitter on $\mathscr{A}$ and α_c, θ_c

Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be the affinities of two inliers estimated during step 1 of Algorithm 3. Due to the presence of point jitter and perspective distortion, they are usually different even if we have used inliers for their computation. As a consequence, θ_c and α_c have to be carefully chosen in order to decide whether two affinities are similar or not. In the following, we only study the influence of point jitter on the difference between affinities since perspective distortion in a small region can be regarded as a special case of point jitter.

As one can expect, the smaller the distance between points composing a Quad-Point, the more impact point jitter has on the computation of the affinities. Moreover, the fewer correspondences we use to estimate an affinity, the less precise the result. Both observations lead to the fact that the difference between two affinities $\mathscr{A}_1$ and $\mathscr{A}_2$ becomes more important when k gets smaller. In other word, $k = 3$ is probabilistically speaking the worst situation where the difference between $\mathscr{A}_1$ and $\mathscr{A}_2$ might be the greatest.

Experimental results are collected from 1000 runs with 200 points uniformly distributed and $k = 3$. Cumulative distributions for α_u, α_v and $\Delta\theta$ are illustrated in Fig. 7 with $\eta = 5\%$. α_u, α_v and $\Delta\theta$ are computed from inliers according to equation (13).

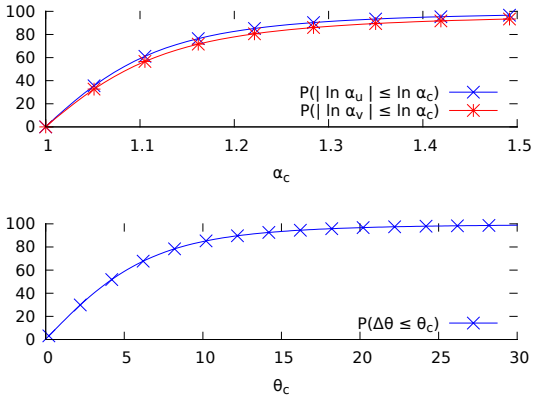


Figure 7: *Cumulative distribution for $P(|\ln\alpha_u| \leq \ln\alpha_c), P(|\ln\alpha_v| \leq \ln\alpha_c)$ (top) and $P(\Delta\theta \leq \theta_c)$ (bottom)*

With $\alpha_c = 1.3$, $\theta_c = 10°$, $P(|\ln\alpha_v| \leq \ln\alpha_c) = 86.8\%$, $P(|\ln\alpha_u| \leq \ln\alpha_c) = 91\%$ and $P(\Delta\theta \leq \theta_c) = 84.7\%$, we show that about $P(|\ln\alpha_v| \leq \ln\alpha_c) \times P(|\ln\alpha_u| \leq \ln\alpha_c) \times P(\Delta\theta \leq \theta_c) = 66.9\%$ of inliers affinity pairs are judged similar in the least favourable case.

6 Results

In this section, we compare the efficiency and the robustness of our technique (RRDM), an algorithm combining local voting (cf. Section 4.2.1) and USAC [Raguram et al. 2013] (USAC) and Random Dot Markers (RDM). An homography is estimated only if at least 9 inliers are found. This value is given by RDM and we apply this rule to all the three methods for a fair comparison.

For RRDM, default parameters are $B = 100, k = 6, D_{max} = 0.1, \eta = 3\%, c_{vote} = 4, \alpha_c = 1.3, \theta_c = 10°$. For USAC, all optimization options and PROSAC-based sampling are selected, "inlierThreshold"

(a threshold of symmetric error) is set to $2\sqrt{2}\sigma$ while other parameters are kept as their default values [Raguram et al. 2013]. For RDM, only "tracking by matching" method (i.e. a matching procedure is applied for each image without taking advantage of results obtained in previous frames) of the original Random Dot Markers with default values for parameters [Uchiyama and Saito 2011] is used for a fair comparison. Experiments are done on a PC with an Intel Xeon E3 1240 @3.40GHz CPU and 16GB of RAM.

6.1 Synthetic images

Synthetic images are used to study expected matching time in function of point quantity, as well as the effects of jitter on three methods. We choose synthetic images because both jitter value and the number of points are difficult to control in real images.

Synthetic images are constructed as a *virtual marker* containing m randomly distributed points in a 1280×720 rectangle. It serves as P_r. This virtual marker is then contaminated by point jitter which is simulated by additive Gaussian noise, and by βm randomly generated noisy points. This contaminated virtual marker is placed before a virtual camera so that the angle between the virtual marker plan and the focal plane of the virtual camera is $30°$. P_i is the camera image of the point set in the contaminated virtual marker. Remember that Gaussian noise is controlled by η (cf. Section 4).

Due to the presence of jitter, an exact estimation is impossible. We use Φ_3 mentioned in [Huynh 2009] to measure if a matching is "precise". With $\mathbf{q_0}$ being the true quaternion of the virtual marker plan and $\mathbf{q}$ the estimated one, a matching is "precise" if:

$$\Phi_3 = \arccos(\|\mathbf{q_0} \cdot \mathbf{q}\|) \leq 1.5° \qquad (14)$$

In the "Point quantity experiment" (cf. Fig. 8), influence of the amount of point on performance of methods are studied with $\beta = 15\%$ and $\eta = 3\%$. Results are averaged from 1000 experiments. It shows that even with a moderate point jitter and a small quantity of extra points, RDM gives a very poor estimation. USAC's estimation is comparable with the one obtained by RRDM but the execution time explodes when m increases due to lower inlier ratios. Our method remains consistently precise starting at $m = 100$ even if RDM remains slightly more efficient than RRDM. This characteristic is very important since for a city like Edinburgh, a map of the city center contains about 200 road intersections. RDM and USAC work best at $m = 100$, so we choose this value for later comparison aiming at a more comparative study.

In the "Jitter experiment" (cf. Fig. 9), LS curve is obtained using classical least square estimation **with known** point correspondence and can be considered as an upper bound. Statistics are obtained from 1000 experiments. It can be seen that RDM and USAC suffer from a very steep performance decrease with η goes big.

6.2 Real markers

We studied the robustness of three methods against extra points, missing points and partial occlusion with real markers. Two kinds of equipment are used: a LCD screen filmed at 1920×1080 by an iPad Air video camera and a A4 paper at 1280×720 filmed with a Logitech C270 camera, as illustrated in Fig. 10. A video sequence of 200 to 300 frames is recorded for each case. A visual assessment is provided for all these experiments after programs give the estimated homography. We have to notice that in these real images, no point jitter occurs, which cannot highlight the robustness of our method. We show that even in situations which are in favor of RDM, our method works better as well.

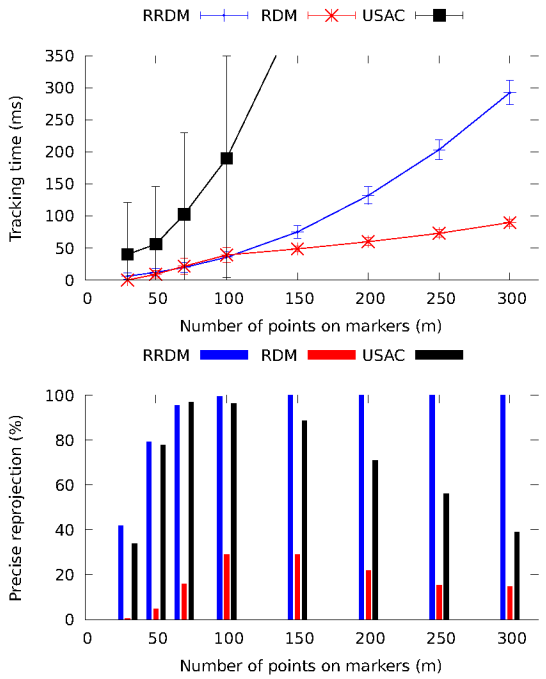


Figure 8: *Point quantity experiment: robustness and speed with respect to m.* $\beta = 15\%, \eta = 3\%$.

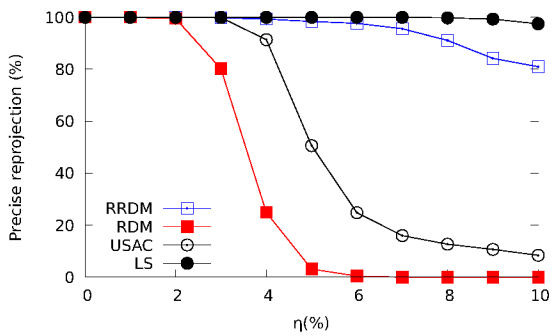


Figure 9: *Jitter experiment:* $m = 100$.

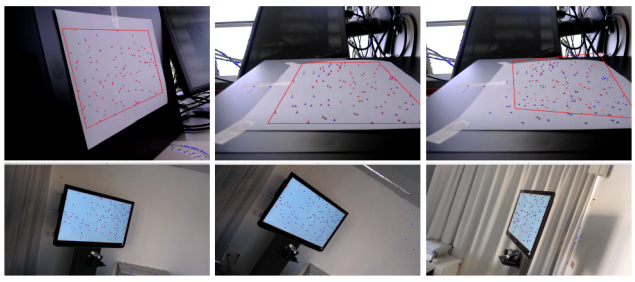


Figure 10: *Video snapshots: upper band for paper markers (the rightest one did not work), lower band for screen-based markers.*

In the "extra point experiment", βm extra points are added inside markers to simulate noisy points. Results are shown in Fig. 11. RDM's performance decreases significantly with respect to our proposal. USAC gives many results when β is high but most of which are wrong estimations.

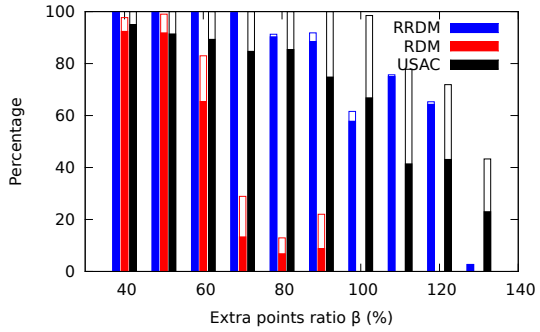


Figure 11: *Extra point experiment, βm extra points are added inside the marker. Solid bars represent correct reprojections, empty bars represent incorrect ones.*

In the "tilted view experiment" (cf. Fig. 12), we compare the efficiency of three methods on the basis of the perspective angle ψ. ψ is defined as the angle between the marker plane and the camera focal plane, so a top view gives $\psi = 0°$. RRDM and USAC give more good estimations than RDM, especially under large perspective angle $\psi = 70°$. At $\psi = 75°$, USAC seems to outperform RRDM, but it has too many bad estimations to be used.

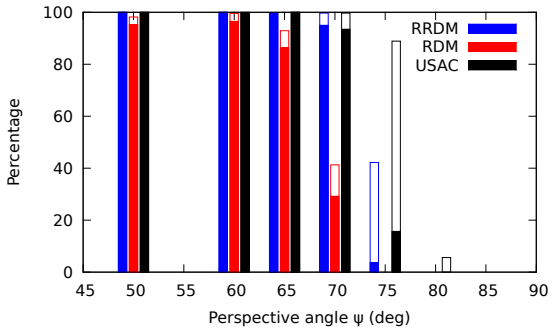


Figure 12: *Tilted view experiment. Solid bars represent correct reprojections, empty bars represent incorrect ones.*

The "missing point experiment" is dedicated to incomplete markers detection and tracking. There can be two kinds of missing elements in markers: (i) a whole part of a marker is invisible (occluded by another real object for example) and (ii) randomly distributed missing points are removed from the marker. Fig. 13 shows the results of experiments where (left) a marker is gradually occluded by a sheet of paper sliding on the screen and (right) where random points are removed from the marker. In both cases, our algorithm behaves slightly better, although it is a bit more significant in the randomly missing points case. The performance decay is logical since all algorithms use neighborhood relationships, which are quickly altered by random removal of points.

6.3 Natural map tracking

We apply RRDM to a map geo-referencing problem in the "map tracking experiment". Three different unprepared printed maps

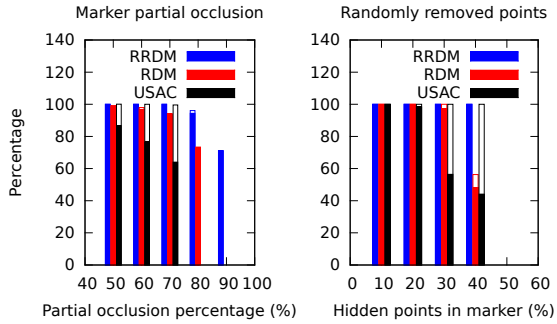


Figure 13: *Missing point experiment. Solid bars represent correct reprojections, empty bars represent incorrect ones.*

Table 1: *Qualities of detections in map tracking experiment (Mean value $\pm$ standard deviation).*

Map ID	P_i size	Noisy points	Inliers found
(a)	236.7 ± 14.7	52.7 ± 5.0	102.0 ± 28.1
(b)	279.6 ± 16.9	114.9 ± 11.9	76.8 ± 19.8
(c)	237.1 ± 10.3	51.7 ± 7.4	56.9 ± 5.5

of downtown Edinburgh containing different textures are used, cf. Fig. 1 (a-c). We manually extracted from map (a) 233 intersections coordinates, which serve as the reference dot pattern (i.e. P_r). For each map, a video sequence containing about 100 frames is recorded with an iPad. Although only one geo-referencing operation is needed for each sequence of video in order to initialize tracking (if no tracking failure appears), we choose to apply it to each frame of the videos to show RRDM's behavior under diverse circumstances. For each frame, an automatic intersection detection technique based color segmentation (similar to [Callier and Saito 2011]) is used to produce P_i on the fly. In this experiment η is set to 5%.

To give an idea of the complexity of the problem we tackle, we manually analyzed the detection results on 10 randomly picked frames for each video (cf. Table 1). We consider a detected point to be a real intersection if it does not lay too far (closer than roughly 3 times the local road width) from any intersection on the map, otherwise it is considered to be a noisy point. However, a real intersection point may have no corresponding point in P_r as well. This can happen because different printed maps may not have the exact same road network. As a consequence, new road intersections may arise and the total number of inliers remains unknown. Detection for map (a) has a fairly good quality thanks to its uniform road color (white) and high image quality (originally 1171×795 pixels). Roads in map (b) are white and yellow, those colors are similar to some background elements, leading to a much noisier detection result. Map (c) is also challenging since its original quality is low (553×461 pixels). Furthermore, scaling is not homogeneous between maps (c) and (a) (remember that P_r was constructed based on map (a)) thus preventing some inliers to be reprojected close enough to their corresponding detected points. This prevents the result list from a significant expansion during the recovery phase (cf. Section 4.2.3), as a consequence, correct reprojections are confined in a subregion of the map (see top left area in Fig.1 (c)).

Tracking results of RRDM for the videos of maps (a), (b) and (c) are presented in Table 2. As both the total number of inliers and the ground truth of the homography are unknown, it is difficult to estimate whether a reprojection is correct. We consider a reprojection to be good if the basement of the 3D model of Edinburgh's castle

Table 2: *Results of map tracking experiment. Good matching rates, detection time and matching time per frame (Mean value ± standard deviation).*

Map ID	Good matching (%)	Detection time (ms)	Matching time (ms)
(a)	100.0%	73.0 ± 11.5	167.0 ± 14.2
(b)	90.2%	210.6 ± 23.8	223.6 ± 18.7
(c)	96.0%	81.2 ± 6.1	182.4 ± 15.4

is inside the castle area on the map which is mainly bounded by *Princes St., Kings Stables Rd, Johnston Terrace* and *The Mount* on the map. Note that neither RDM nor USAC work for any of videos.

Finally, to illustrate the pertinence of our approach, GIS data of the city of Edinburgh extracted from OpenStreetMap[1], is used to compute coordinates of actual road intersections in the vicinity of our map. Those coordinates are computed from a GIS and obtained from a spatial database engine (H2GIS[2]) and used as ground truth for our point pattern matching. Intersections detected in real-time from the unprepared paper map are thus matched with the intersections extracted from the GIS to compute the camera pose. Furthermore, in order to evaluate the precision of our matching, we display onto the map the road network which is also extracted from the GIS. Fig. 1 (d) illustrates the results of this approach.

7 Conclusion

In this paper, we have introduced RRDM, a method to robustly track random dot markers, where over and under point detection, large perspective distortion and inaccurate detection (point jitter) can be taken into account. Unlike traditional feature-point methods, RRDM is also able to robustly track 2D textureless object. Recommendations on parameters are also given in order to facilitate user's implementation.

Synthetic images are used to demonstrate the robustness of RRDM against point jitter and when the marker is composed of a large number of points. In those cases, RRDM clearly outperforms RDM. We also proved through experiments with real videos that RRDM is more robust than RDM in presence of partial occlusion, randomly missing pattern points or under large perspective distortion, and especially with extra points in the pattern. Regarding efficiency, a real-time tracking (30 Hz) can be achieved when the number of points in a marker is moderate (about 100 points).

Finally, as an application, we applied RRDM on registration of unprepared printed maps where purely geometric coordinates of road intersections (extracted from a GIS) were used as the point pattern. This pattern was then matched against road intersections detected from the maps automatically and in real-time. At present, this approach (automated real-time intersection detection combined with RRDM) is not fast enough to be used in this application for real-time tracking, but can be used as an initialization step. As for now, a color filter is generated offline for each map before detecting in real-time the intersections. Our future work will focus on high quality "filterless" detection of road intersections as well as on optimizing the bottlenecks of out method, namely the local voting (4.2.1) and rejection (4.2.2) stages.

[1]The GIS data can be found here: `http://www.sharegeo.ac.uk/handle/10672/238`

[2]`http://www.h2gis.org`

8 Appendix

We want to demonstrate that two Quad-Points are affinely equivalent if and only if their TSR descriptors are equal, which means:

$$\mathbf{t}(Q) = \mathbf{t}(Q') \Leftrightarrow Q \equiv Q' \tag{15}$$

As Q and Q' have the same affine invariants, the backward demonstration is direct from equations (1) and (2). We focus on the forward demonstration.

We first show that for a Quad-Point (Q), the position of point $F(u_F, v_F)$ is uniquely defined by the positions of M, L, R and $t(Q) = (X, Y, c)$. Assume M to be at $(0,0)$. $(u_L, v_L), (u_R, v_R)$ are positions of L, R after translation respectively.

The 2D plane is divided into seven regions (cf. Fig. 14). According to our definition of TSR (Section 3.1), F cannot lay in ③ since it would be a "left point" rather than a "face point". For the same reason ④ $\sim$ ⑥ does not fit. So, F can only lay in ① (c=1),② (c=2) and ⑦ (c=3).

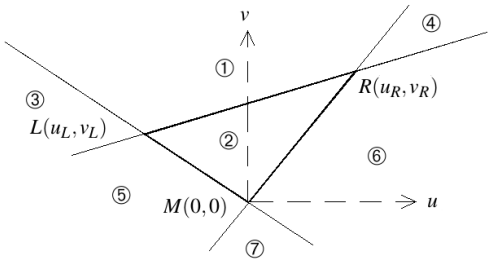


Figure 14: *Possible regions for $F(u_F, v_F)$*

For $c = 1$:

$$\begin{aligned} S_a &= \frac{1}{2}\begin{vmatrix} u_R & u_F \\ v_R & v_F \end{vmatrix} + \frac{1}{2}\begin{vmatrix} u_F & u_L \\ v_F & v_L \end{vmatrix} \\ S_{FLR} &= \frac{1}{2}\begin{vmatrix} u_R - u_L & u_F - u_L \\ v_R - v_L & v_F - v_L \end{vmatrix} \\ S_{MFL} &= \frac{1}{2}\begin{vmatrix} u_F & u_L \\ v_F & v_L \end{vmatrix} \end{aligned} \tag{16}$$

With (2), we have:

$$\begin{aligned} &\mathbb{Z}\begin{pmatrix} u_F \\ v_F \end{pmatrix} = \begin{pmatrix} -u_R v_L + v_R u_L \\ 0 \end{pmatrix} \\ &\text{with } \mathbb{Z} = \begin{bmatrix} (1-X)(v_R - v_L) & -(1-X)(u_R - u_L) \\ -Y v_R - (1-Y) v_L & Y u_R + (1-Y) u_L \end{bmatrix} \end{aligned} \tag{17}$$

This equation gives a unique solution when $|\mathbb{Z}| \neq 0$, meaning that the coordinates of F are uniquely defined.

$$|\mathbb{Z}| = -(1-X)\begin{vmatrix} u_R & u_L \\ v_R & v_L \end{vmatrix} = -2(1-X) S_{MLR} \neq 0 \tag{18}$$

A similar approach can be applied for $c = 2$ and $c = 3$ for which $|\mathbb{Z}| = 2(1+Y) S_{MLR}$ and $|\mathbb{Z}| = 2X S_{MLR}$ respectively and where $|\mathbb{Z}|$ cannot be zero.

The forward demonstration is as follows: Assume that $Q =< M; F, R, L >$ and $Q' =< M'; F', R', L' >$ have the same TSR descriptor (X, Y, c). Calculate the affine transformation $\mathscr{A}$ which makes $M' = \mathscr{A}(M)$, $R' = \mathscr{A}(R)$, $L' = \mathscr{A}(L)$. If $\mathscr{A}(F) = F^*$, then $Q^* =< M'; F^*, R', L' > \equiv Q'$. Since Q and Q^* share the same TSR descriptor and $< M, R, L, F = F^* >$ is unique. $Q \equiv Q'$.

References

AIGER, D., AND KEDEM, K. 2010. Approximate Input Sensitive Algorithms for Point Pattern Matching. *Pattern Recogn. 43*, 1 (Jan.), 153–159.

BALLARD, D. H. 1981. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recogn. 13*, 2 (Jan.), 111–122.

BAY, H., ESS, A., TUYTELAARS, T., AND VAN GOOL, L. 2008. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst. 110*, 3 (June), 346–359.

CALLIER, S., AND SAITO, H. 2011. Automatic Road Extraction from Printed Maps. In *Proceedings of the IAPR Conference on Machine Vision Applications*, 243–246.

CALONDER, M., LEPETIT, V., OZUYSAL, M., TRZCINSKI, T., STRECHA, C., AND FUA, P. 2012. BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Trans. Pattern Anal. Mach. Intell. 34*, 7, 1281–1298.

CHUM, O., AND MATAS, J. 2005. Matching with PROSAC - Progressive Sample Consensus. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, CVPR '05, 220–226.

DATTA, A., KONG, A. W.-K., GHOSH, S., AND TRAU, D. 2013. A Fast Point Pattern Matching Algorithm for Robust Spatially Addressable Bead Encoding. In *Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on*, 1–7.

FISCHLER, M. A., AND BOLLES, R. C. 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM 24*, 6 (June), 381–395.

GATRELL, L. B., HOFF, W. A., AND SKLAIR, C. W. 1992. Robust Image Features: Concentric Contrasting Circles and Their Image Extraction. In *Proceedings of SPIE, Cooperative Intelligent Robotics in Space II*, vol. 1612, 235–244.

HARTLEY, R. I., AND ZISSERMAN, A. 2004. *Multiple View Geometry in Computer Vision*, 2 ed. Cambridge University Press, ISBN: 0521540518, New York, NY, USA.

HUYNH, D. Q. 2009. Metrics for 3D Rotations: Comparison and Analysis. *J. Math. Imaging Vis. 35*, 2 (Oct.), 155–164.

JIANG, X., BROELEMANN, K., WACHENFELD, S., AND KRÜGER, A. 2011. Graph-based markerless registration of city maps using geometric hashing. *Comput. Vis. Image Underst. 115*, 7 (July), 1032–1043.

KATO, H., AND BILLINGHURST, M. 1999. Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, IEEE Computer Society, 85–94.

LEPETIT, V., AND FUA, P. 2005. Monocular Model-based 3D Tracking of Rigid Objects. *Found. Trends. Comput. Graph. Vis. 1*, 1 (Jan.), 1–89.

LOWE, D. G. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision 60*, 2 (Nov.), 91–110.

MARTEDI, S., AND SAITO, H. 2011. Augmented Fly-through using Shared Geographical Data. In *21st International Conference on Artificial Reality and Telexistence*, ICAT 2011, 47–52.

MORRISON, A., MULLONI, A., LEMMELÄ, S., OULASVIRTA, A., JACUCCI, G., PELTONEN, P., SCHMALSTIEG, D., AND REGENBRECHT, H. 2011. Mobile Augmented Reality: Collaborative Use of Mobile Augmented Reality with Paper Maps. *Comput. Graph. 35*, 4, 789–799.

NAKAI, T., KISE, K., AND IWAMURA, M. 2006. Use of Affine Invariants in Locally Likely Arrangement Hashing for Camera-based Document Image Retrieval. In *Proceedings of the 7th International Conference on Document Analysis Systems*, Springer-Verlag, Berlin, Heidelberg, DAS'06, 541–552.

OLSON, C. F. 1997. Efficient Pose Clustering Using a Randomized Algorithm. *Int. J. Comput. Vision 23*, 2 (June), 131–147.

RAGURAM, R., CHUM, O., POLLEFEYS, M., MATAS, J., AND FRAHM, J.-M. 2013. USAC: A Universal Framework for Random Sample Consensus. *IEEE Trans. Pattern Anal. Mach. Intell. 35*, 8 (Aug.), 2022–2038.

REILLY, D., RODGERS, M., ARGUE, R., NUNES, M., AND INKPEN, K. 2006. Marked-up Maps: Combining Paper Maps and Electronic Information Resources. *Personal Ubiquitous Comput. 10*, 4 (Mar.), 215–226.

REITMAYR, G., EADE, E., AND DRUMMOND, T. 2005. Localisation and Interaction for Augmented Maps. In *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*, IEEE Computer Society, Washington, DC, USA, ISMAR '05, 120–129.

REKIMOTO, J. 1998. Matrix: A Realtime Object Identification and Registration Method for Augmented Reality. In *Proceedings of the Third Asian Pacific Computer and Human Interaction*, IEEE Computer Society, Washington, DC, USA, APCHI '98, 63–68.

SCHÖNING, J., KRÜGER, A., AND MÜLLER, H. 2006. Interaction of mobile camera devices with physical maps. In *Adjunct Proceeding of the Fourth International Conference on Pervasive Computing*, 121–124.

UCHIYAMA, H., AND MARCHAND, E. 2011. Toward Augmenting Everything: Detecting and Tracking Geometrical Features on Planar Objects. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, 17–25.

UCHIYAMA, H., AND SAITO, H. 2011. Random Dot Markers. In *Proceedings of the 2011 IEEE Virtual Reality Conference*, IEEE Computer Society, Washington, DC, USA, VR '11, 271–272.

UCHIYAMA, H., SAITO, H., SERVIÈRES, M., AND MOREAU, G. 2009. AR GIS on a Physical Map based on Map Image Retrieval using LLAH Tracking. In *IAPR Conference on Machine Vision Application*, MVA'09, 382–385.

UCHIYAMA, H., SAITO, H., SERVIÈRES, M., AND MOREAU, G. 2011. Camera tracking by online learning of keypoint arrangements using LLAH in augmented reality applications. *Virtual Real. 15*, 2-3, 109–117.

VAN WAMELEN, P., LI, Z., AND IYENGAR, S. 2004. A fast expected time algorithm for the 2-D point pattern matching problem. *Pattern Recogn. 37*, 8 (Aug.), 1699–1711.

WAGNER, D., AND SCHMALSTIEG, D. 2007. ARToolKitPlus for Pose Tracking on Mobile Devices. In *Proceedings of 12th Computer Vision Winter Workshop*, CVWW '07, 139–146.

WOLFSON, H. J., AND RIGOUTSOS, I. 1997. Geometric Hashing: An Overview. *IEEE Comput. Sci. Eng. 4*, 4 (Oct.), 10–21.