

# Least squares contour alignment

Ivan Markovsky and Sasan Mahmoodi

**Abstract**—The contour alignment problem, considered in this paper, is to compute the minimal distance in a least squares sense, between two explicitly represented contours, specified by corresponding points, after arbitrary rotation, scaling, and translation of one of the contours. This is a constrained nonlinear optimization problem with respect to the translation, rotation and scaling parameters, however, it is transformed into an equivalent linear least squares problem by a nonlinear change of variables. Therefore, a global solution of the contour alignment problem can be computed efficiently. It is shown that a normalized minimum value of the cost function is invariant to ordering and affine transformation of the contours and can be used as a measure for the distance between the contours. A solution is proposed to the problem of finding a point correspondence between the contours.

**Index Terms**—Contour alignment, image registration, translation, rotation, scaling, affine invariance, least squares.

**EDICS category:** IMD-PATT, IMD-ANAL

## I. PROBLEM FORMULATION

Consider two contours  $\mathcal{C}_1$  and  $\mathcal{C}_2$  in  $\mathbb{R}^2$ , specified by  $N$  corresponding points

$$p^{(i)} \in \mathcal{C}_1 \subset \mathbb{R}^2 \leftrightarrow q^{(i)} \in \mathcal{C}_2 \subset \mathbb{R}^2, \quad i = 1, \dots, N.$$

In what follows, we will use the matrices of the stacked next to each other points

$$C_1 := [p^{(1)} \ \dots \ p^{(N)}] \quad \text{and} \quad C_2 := [q^{(1)} \ \dots \ q^{(N)}].$$

( $A := B$  and  $B := A$  mean that  $A$  is by definition equal to  $B$ .) Let  $\mathcal{R}_\theta$  be the operator that rotates by  $\theta \in [-\pi, \pi)$  rad (positive angle corresponding to anticlockwise rotation), i.e.,

$$\mathcal{R}_\theta(p) := \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} p$$

and let  $\mathcal{A}_{a,\theta,s}$  be the operator that rotates by angle  $\theta \in [-\pi, \pi)$  rad, scales by factor  $s > 0$ , and translates by vector  $a \in \mathbb{R}^2$ , i.e.,

$$\mathcal{A}_{a,\theta,s}(p) = s\mathcal{R}_\theta(p) + a.$$

Acting on the matrix  $C_1$ ,  $\mathcal{A}_{a,\theta,s}$  transforms each column  $p^{(i)}$  of  $C_1$ . The “size” of the contour  $\mathcal{C}_1$  is measured by the Frobenius norm  $\|C_1\|_F := \sqrt{\sum_{i=1}^N \|p^{(i)}\|_2^2}$ .

The considered least squares contour alignment problem is defined as

$$\text{minimize over } a \in \mathbb{R}^2, s > 0, \theta \in [-\pi, \pi) \|C_1 - \mathcal{A}_{a,\theta,s}(C_2)\|_F. \quad (1)$$

Because of inequality constraints on  $\theta, s$  and nonlinear dependence of  $\mathcal{A}_{a,\theta,s}(\cdot)$  on the optimization variables  $\theta$  and  $s$ , (1) is a constrained nonlinear least squares optimization problem. This nonlinear minimization scheme is widely employed in

the literature in an explicit (see, e.g., [1], [2], [3], [4], [5]) and implicit (see, e.g., [6], [7], [8]) contour representation frameworks. Local optimization methods are used for solving the alignment problems, however, they require initial approximation and do not give guarantee that a globally optimal solution is computed. In addition, they may have convergence problems and be computationally expensive.

The main contribution of this paper, presented in Section II, is a (nonlinear) change of variables that transforms problem (1) to a linear least squares problem. A global minimum point of (1) is therefore computable by standard numerical linear algebra methods. Moreover, the computational complexity of the resulting method is linear with respect to the number of points  $N$ . These are major advantages of the proposed contour alignment method over the ones of [6], [7]. In addition, as a by product of the solution of problem (1), we define in Section III an affine invariant distance measure between contours. (In comparison, a distance measure defined in [7] is translation and scale invariant and requires a data preprocessing step.)

The main disadvantage of using problem (1) in practical computer vision problems is that it requires corresponding points from the contours to be specified. Such points may not be available in practice. In order to address this issue, in Section IV we propose an extension of the method for finding point correspondence. The extended method requires solution of a sequence of least squares alignment problems and has quadratic computational complexity in  $N$ . As shown in Section V,  $O(N^2)$  computationally complexity is still feasible for realistic registration problems.

## II. MAIN RESULT

**Theorem 1.** *Problem (1) is equivalent to the following least squares problem*

$$\text{minimize over } (a_1, a_2, b_1, b_2) \in \mathbb{R}^4 \left\| \begin{bmatrix} p_1^{(1)} \\ p_2^{(1)} \\ \vdots \\ p_1^{(N)} \\ p_2^{(N)} \end{bmatrix} - \begin{bmatrix} 1 & 0 & q_1^{(1)} & -q_2^{(1)} \\ 0 & 1 & q_2^{(1)} & q_1^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & q_1^{(N)} & -q_2^{(N)} \\ 0 & 1 & q_2^{(N)} & q_1^{(N)} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix} \right\|_2, \quad (2)$$

where the relation between the parameters  $b_1, b_2$  in (2) and the parameters of  $\theta, s$  in (1) is given by

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = s \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}, \quad \begin{bmatrix} \theta \\ s \end{bmatrix} = \begin{bmatrix} \sin^{-1}(b_2 / \sqrt{b_1^2 + b_2^2}) \\ \sqrt{b_1^2 + b_2^2} \end{bmatrix}. \quad (3)$$

*Proof:* Consider a point  $q \in \mathbb{R}^2$  and define

$$q_r := \mathcal{R}_{\theta_r}(q).$$

Any angle  $\theta_r \in (-\pi, \pi)$ ,  $\theta_r \neq 0$  can be used, however, the change of variables is particularly simple for  $\theta_r = \pm\pi/2$ . The key observation is that

$$\mathcal{A}_{a,\theta,s}(q) = \begin{bmatrix} q & q_r \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + a,$$

where  $b_1, b_2$  and  $\theta, s$  are in a one-to-one relation that is derived by solving the equation

$$\begin{bmatrix} q & q_r \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = s \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} q \quad (4)$$

for  $b_1$  and  $b_2$ , given  $s$  and  $\theta$ , and for  $\theta$  and  $s$ , given  $b_1$  and  $b_2$ . In the case of  $\theta_r = \pi/2$ , the solution of (4), i.e., the relation between the original and transformed parameters is (3).

In terms of the parameters  $a_1, a_2, b_1, b_2$ , (1) is reduced to a linear least squares problem

$$\text{minimize} \left\| \begin{bmatrix} p_1^{(1)} \\ p_2^{(1)} \\ \vdots \\ p_1^{(N)} \\ p_2^{(N)} \end{bmatrix} - \begin{bmatrix} 1 & 0 & q_1^{(1)} & q_r^{(1)} \\ 0 & 1 & q_2^{(1)} & q_r^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & q_1^{(N)} & q_r^{(N)} \\ 0 & 1 & q_2^{(N)} & q_r^{(N)} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix} \right\|_2. \quad (5)$$

In the case of  $\theta_r = \pi/2$ , (5) simplifies to (2).  $\square$

*Example 2.* We illustrate Theorem 1 on the example in Fig. 1 (left), where the ‘‘correct’’ solution is obvious by eye inspection and matches the computed solution, shown in Fig. 1 (right).

*Example 3.* In order to illustrate the computational efficiency of the proposed contour alignment algorithm, we show in Fig. 2 the computation time, for an implementation in MATLAB 7.3, run on a PC with 2.13GHz CPU, as a function of the number of points  $N$  for randomly generated data. E.g., a problem with one million points is solved in about 0.5 sec.

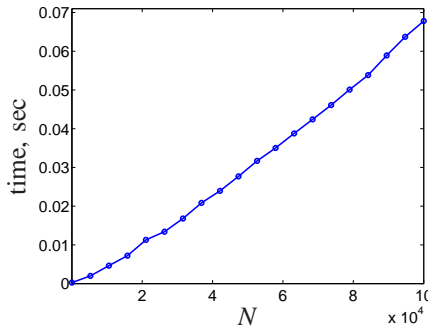


Fig. 2. Computation time for solving problem (2) with random data.

### III. DISTANCE BETWEEN CONTOURS

It is tempting to think of the minimum value of (1)

$$d'(C_1, C_2) := \min_{a \in \mathbb{R}^2, s > 0, \theta \in [-\pi, \pi]} \|C_1 - \mathcal{A}_{a,\theta,s}(C_2)\|_F$$

as a distance measure between the contours  $\mathcal{C}_2$  and  $\mathcal{C}_1$  modulo rotation, scaling, and translation. In general, however,  $d'(C_1, C_2) \neq d'(C_2, C_1)$ , so that  $d'$  is not a proper distance measure. In addition,  $d'(C_1, C_2)$  is not invariant to simultaneous

affine transformation of the contours  $\mathcal{C}_1$  and  $\mathcal{C}_2$ , which is an undesirable feature in most computer vision application.

In [7] a related distance measure (for implicitly represented contours) is made translation and scale invariant by centering and normalization of one of the contours. The centering and normalization operations can be viewed as a preprocessing step. For the least squares alignment problem in this paper the following result holds.

**Proposition 4.** *If the contours  $\mathcal{C}_1$  and  $\mathcal{C}_2$  are centered, i.e., with  $\mathbf{1}_N := \text{col}(1, \dots, 1) \in \mathbb{R}^N$*

$$C_1 \mathbf{1}_N = C_2 \mathbf{1}_N = 0, \quad (6)$$

*then  $d'(C_1, C_2)$  is rotation-invariant, i.e.,*

$$d'(C_1, C_2) = d'(\mathcal{R}_\theta(C_1), \mathcal{R}_\theta(C_2)), \text{ for any } \theta \in [-\pi, \pi]. \quad (7)$$

*If, in addition,  $\mathcal{C}_1$  and  $\mathcal{C}_2$  are normalized, i.e.,*

$$\|C_1\|_F = \|C_2\|_F = 1, \quad (8)$$

*then*

$$d'(C_1, C_2) = d'(C_2, C_1). \quad (9)$$

*Proof:* The proof is given in the appendix.

*Example 5.* Consider again the contours  $\mathcal{C}_1$  and  $\mathcal{C}_2$  from Example 2. Now, we preprocess the points  $p^{(i)}$  and  $q^{(i)}$ , so that the resulting contours, say  $\mathcal{C}_{1,c}$  and  $\mathcal{C}_{2,c}$ , are centered. As a numerical verification of (7), we have

$$d'(C_{1,c}, C_{2,c}) = d'(\mathcal{R}_{0.3}(C_{1,c}), \mathcal{R}_{0.3}(C_{2,c})) = 0.2626.$$

Let, in addition, preprocess the points  $p^{(i)}$  and  $q^{(i)}$ , so that the resulting contours, say  $\mathcal{C}_{1,cn}$  and  $\mathcal{C}_{2,cn}$ , are normalized, according to (6) and (8). As a numerical verification of (9), we have

$$d'(C_{1,cn}, C_{2,cn}) = d'(C_{2,cn}, C_{1,cn}) = 0.083.$$

Our next result shows that a small modification of  $d'$ —normalization by the size of the centered contour  $\mathcal{C}_1$ —is affine invariant and independent of the ordering of the contours. An alternative view of the result is that the preprocessing step is built in the definition of the new distance measure.

**Definition 6** (2-norm distance between  $\mathcal{C}_1$  and  $\mathcal{C}_2$  modulo affine transformation).

$$d(C_1, C_2) := \frac{1}{\|C_1 - \frac{1}{N} C_1 \mathbf{1}_N \mathbf{1}_N^\top\|_F} \times \min_{a \in \mathbb{R}^2, s > 0, \theta \in [-\pi, \pi]} \|C_1 - \mathcal{A}_{a,\theta,s}(C_2)\|_F. \quad (10)$$

Note that the matrix  $C_{1,c} := C_1 - \frac{1}{N} C_1 \mathbf{1}_N \mathbf{1}_N^\top$  corresponds to the centered contour  $\mathcal{C}_{1,c}$ .

**Theorem 7.**  *$d(C_1, C_2)$  is symmetric and affine invariant, i.e.,*

$$d(C_1, C_2) = d(C_2, C_1) = d(\mathcal{A}_{a,\theta,s}(C_1), \mathcal{A}_{a,\theta,s}(C_2)), \text{ for all } a \in \mathbb{R}^2, \theta \in [-\pi, \pi], \text{ and } s > 0. \quad (11)$$

*Proof:* The proof of Theorem 7 uses the same technique as the one used in the proof of Proposition 4 and is skipped.

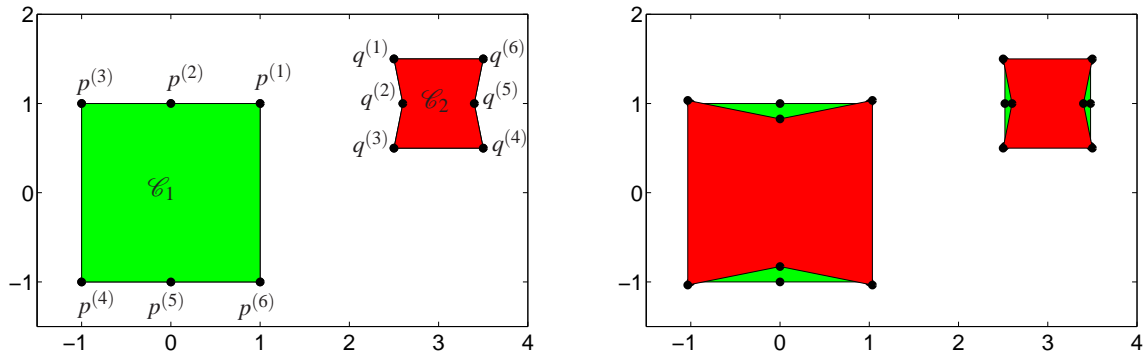


Fig. 1. Left: Contours  $\mathcal{C}_1$  and  $\mathcal{C}_2$  with 6 corresponding points  $p^{(i)} \leftrightarrow q^{(i)}$ . Right: optimal alignment of  $\mathcal{C}_2$  to  $\mathcal{C}_1$  and  $\mathcal{C}_1$  to  $\mathcal{C}_2$ .

Note that  $d'(C_1, C_2)$  is equal to the *absolute size* of the difference  $C_1 - \mathcal{A}_{a,\theta,s}(C_2)$ , while  $d(C_1, C_2)$  is equal to the size of the difference *relative to the size of  $C_{1,c}$* . In particular, for  $d'(C_1, C_2) \neq 0$ , it is not possible to decide how “far” is  $\mathcal{C}_1$  from  $\mathcal{C}_2$ , while  $100 \times d(C_1, C_2)$  can be interpreted as a “percentage difference”.

*Example 8.* For the contours  $\mathcal{C}_1$  and  $\mathcal{C}_2$  in Example 2, we have  $d'(C_1, C_2) = 0.2626$  and  $d(C_2, C_1) = 0.1265$ , while  $d(C_1, C_2) = d(C_2, C_1) = 0.083$ , i.e., the difference of  $\mathcal{C}_1$  from  $\mathcal{C}_2$  is approximately 8% of the size of  $C_{1,c}$  (or, alternatively, the difference of  $\mathcal{C}_2$  from  $\mathcal{C}_1$  is approximately 8% of  $\|C_{2,c}\|_F$ ).

#### IV. FINDING POINT CORRESPONDENCE BETWEEN CONTOURS

A limitation of problem (1) for application in computer vision is the assumption that the given points  $p^{(i)}$  and  $q^{(i)}$  are corresponding points. Using standard segmentation algorithms, it is more realistic to obtain *sequential* but not necessarily corresponding points from the contours. Corresponding points can, however, be found from given sequential points from  $\mathcal{C}_1$  and  $\mathcal{C}_2$  by solving  $N$  (the number of points representing the contours) least squares problems and selecting the minimum of the  $N$  least squares residuals norms. The procedure is based on shifting the points on one of the contours, say  $\mathcal{C}_1$ , and computing the distance modulo rotation, scaling, and translation from the shifted points of  $\mathcal{C}_1$  to given points of  $\mathcal{C}_2$ . The proposed procedure of finding corresponding points is

$$\begin{aligned} & \text{minimize} && \text{over } k = 1, 2, \dots, N && d(\text{shift}_k(C_1), C_2) && (12) \\ & \text{where} && \text{shift}_k([p^{(1)} & p^{(2)} & \dots & p^{(N)}]) := \\ & && [p^{(k)} & p^{(k+1)} & \dots & p^{(N)} & p^{(1)} & p^{(2)} & \dots & p^{(k-1)}]. \end{aligned}$$

#### V. NUMERICAL EXAMPLE

In order to illustrate how (12) can be applied to find point correspondence in a practical problem, we take two images from the example in [6, page 139, Fig. 1].



First, the edges of the binary images are detected using an edge detection algorithm (e.g., the Prewitt algorithm) in order to obtain a binary edge map. Second, an edge follower starting from an arbitrary point on the edge and finding the nearest neighbor to the current point is applied to the binary edge map to sequentially store the coordinates of the edge points. Finally, a cubic spline interpolation is used to down/up sample the contours to the same number of points (in this example, 300 points).

As a result of the segmentation step, we obtain contours, say  $\mathcal{C}_1$  and  $\mathcal{C}_2$ , which are specified by 300 sequential but not necessarily corresponding points. Then, we solve (12) for  $\mathcal{C}_1$  and  $\mathcal{C}_2$  in order to find the optimal shift. The plot of  $d(\text{shift}_d(C_1), C_2)$  as a function of  $k$  is shown in the left plot of Fig. 3. It takes less than 0.1 seconds in MATLAB version 7.3, run on a PC with 2.13GHz CPU to evaluate the cost function of (12) 300 times (i.e., to solve 300 times least squares problems for the computation of the distance with shifts  $k = 1, 2, \dots, 300$ ).

The optimal shift is found to be  $k^* = 283$  with a corresponding cost function value 0.2149 (indicated in the left plot of Fig. 3 by **X**). The two given contours and the best matching contour  $\mathcal{A}_{a^*,\theta^*,s^*}(C_2)$  (corresponding to the shift  $k = k^*$ ) are shown in the right plot of Fig. 3. The **Xs** indicate the first points on the contours.

MATLAB files reproducing the results presented in the paper are available from:

<http://users.ecs.soton.ac.uk/im/dist.tgz>

#### VI. CONCLUSIONS

We have shown that by using a nonlinear change of variables, the least squares contour alignment problem (1) is solved by the linear least squares method. The implication of

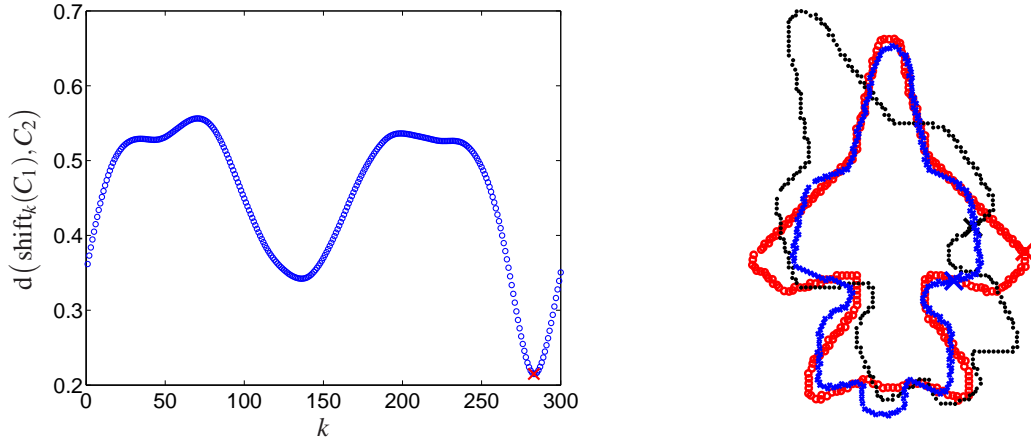


Fig. 3. Left: cost function of (12), Right:  $\mathcal{C}_1$  (circles),  $\mathcal{C}_2$  (dots), and  $\mathcal{A}_{a^*, \theta^*, s^*}(\mathcal{C}_2)$  (crosses); the X's are the first points.

this result is that the problem can be solved globally and efficiently. As a by product of the alignment problem solution, we obtain a distance measure between contours modulo rotation, scaling, and translation that is affine invariant and independent from the ordering of the contours. We also presented a solution to the problem of finding a point correspondence between contours with sequential points. The numerical example shows that the proposed method is an effective solution to the image registration problem and the computational algorithm is robust and efficient.

#### APPENDIX

Let  $\text{vec}(\cdot)$  be the column-wise matrix vectorization operator and  $\otimes$  the Kronecker product. Define

$$\mathbf{p} := \text{vec}(C_1) = \text{col}(p_1^{(1)}, p_2^{(1)}, \dots, p_1^{(N)}, p_2^{(N)})$$

and

$$\mathbf{I} := \mathbf{1}_N \otimes I_2 = \mathbf{1}_N \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Similarly, let  $\mathbf{q} := \text{vec}(C_2)$ ,  $\mathbf{q}_r := \text{vec}(\mathcal{R}_{\pi/2}(C_2))$ , and  $\mathbf{p}_r := \text{vec}(\mathcal{R}_{\pi/2}(C_1))$ . With this notation,  $d'(C_1, C_2)$  is equal to

$$\min_{x \in \mathbb{R}^4} \left\| \mathbf{p} - \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{q} & \mathbf{q}_r \end{bmatrix} x}_{M_q} \right\|_2 = \sqrt{\mathbf{p}^\top M_q (M_q^\top M_q)^{-1} M_q^\top \mathbf{p}}$$

and  $d'(C_2, C_1)$  is equal to

$$\min_{x \in \mathbb{R}^4} \left\| \mathbf{q} - \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{p} & \mathbf{p}_r \end{bmatrix} x}_{M_p} \right\|_2 = \sqrt{\mathbf{q}^\top M_p (M_p^\top M_p)^{-1} M_p^\top \mathbf{q}}.$$

It is easy to see that

$$C_1 \mathbf{1}_N = 0 \iff \mathbf{p}^\top \mathbf{I} = 0.$$

Similarly, from  $C_1 \mathbf{1}_N = 0$ , see (6), it follows that  $\mathbf{q}^\top \mathbf{I} = 0$ . From (6) and the definition of  $\mathbf{p}_r$  and  $\mathbf{q}_r$ , it follows that  $\mathbf{p}_r^\top \mathbf{I} = 0$  and  $\mathbf{q}_r^\top \mathbf{I} = 0$ . Finally, since  $\mathbf{p}_r$  and  $\mathbf{q}_r$  are obtained from  $C_1$  and  $C_2$ , respectively, by rotation by  $\pi/2$  rad, it follows that  $\mathbf{p}_r^\top \mathbf{p} = \mathbf{q}_r^\top \mathbf{q} = 0$ . Using the above identities and (8), we have

$$M_q^\top M_q = M_p^\top M_p = \begin{bmatrix} NI_2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad M_q^\top \mathbf{p} = M_p^\top \mathbf{q} = \begin{bmatrix} 0 \\ \mathbf{q}^\top \mathbf{p} \\ 0 \end{bmatrix}.$$

This proves (9).

In order to prove (7), define

$$\mathbf{p}' := \text{vec}(\mathcal{R}_\theta(C_1)), \quad \mathbf{p}_r' := \text{vec}(\mathcal{R}_{\theta+\pi/2}(C_1))$$

and similarly  $\mathbf{q}'$  and  $\mathbf{q}_r'$ . With this notation,

$$\begin{aligned} d'(\mathcal{R}_\theta(C_1), \mathcal{R}_\theta(C_2)) &= \min_{x \in \mathbb{R}^4} \left\| \mathbf{p}' - \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{q}' & \mathbf{q}_r' \end{bmatrix} x}_{M_q'} \right\|_2 \\ &= \sqrt{\mathbf{p}'^\top M_q' (M_q'^\top M_q')^{-1} M_q'^\top \mathbf{p}'}. \end{aligned}$$

Now, using

$$\mathbf{q}'^\top \mathbf{I} = \mathbf{p}'^\top \mathbf{I} = 0, \quad \mathbf{q}'^\top \mathbf{p} = \mathbf{q}_r'^\top \mathbf{p}_r',$$

(the second identity follows from the property of the multiplication by a rotation matrix to preserve the inner product) and the above identities, it follows that

$$M_q'^\top M_q' = M_q'^\top M_q = \begin{bmatrix} NI_2 & 0 & 0 \\ 0 & \|\mathbf{q}\|_2^2 & 0 \\ 0 & 0 & \|\mathbf{q}\|_2^2 \end{bmatrix}$$

and  $M_q'^\top \mathbf{p} = M_q'^\top \mathbf{p}' = \text{col}(0, \mathbf{q}'^\top \mathbf{p}, 0)$ . This proves (7).

#### REFERENCES

- [1] D. Cremers, F. Tischhauser, J. Weickert, and C. Schnorr, "Diffusion snakes: Introducing statistical shape knowledge into the Mumford-Shah functional," *Int. J. of Computer Vision*, vol. 50, pp. 295–313, 2002.
- [2] T. Cootes, C. Taylor, D. Cooper, and J. Graha, "Active shape models — their training and application," *Computer Vision and Image Understanding*, vol. 61, pp. 38–59, 1995.
- [3] J. Marques and A. Abrantes, "Shape alignment—optimal initial point and pose estimation," *Pattern Recognition Letters*, vol. 18, pp. 49–53, 1997.
- [4] J. Marques, "A fuzzy algorithm for curve and surface alignment," *Pattern Recognition Letters*, vol. 19, pp. 797–803, 1998.
- [5] S. Mahmoodi, B. Sharif, E. Chester, J. Owen, and R. Lee, "Skeletal growth estimation using radiographic image processing and analysis," *IEEE Trans. Information Technology in Biomedicine*, vol. 4, pp. 292–297, 2000.
- [6] A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, W. Grimson, and A. Willsky, "A shape-based approach to the segmentation of medical imagery using level sets," *IEEE Trans. on Medical Imaging*, vol. 22, pp. 137–154, 2003.
- [7] D. Cremers, S. Osher, and S. Soatto, "Kernel density estimation and intrinsic alignment for shape priors in level set segmentation," *Int. J. of Computer Vision*, vol. 69, pp. 335–351, 2006.
- [8] N. Paragios, M. Rousson, and V. Ramesh, "Non-rigid registration using distance functions," *Computer Vision and Image Understanding*, vol. 89, pp. 142–165, 2003.