




REFORM: Rotor Estimation From Object Resampling and Matching

H. Hadfield* , J. Lasenby, M. Ramage and C. Doran

Abstract. In this paper we tackle the problem of correspondence and rotor estimation between models composed of geometric primitives of different types. We frame this problem as searching for the rotor that takes a query model to a reference model. The situations that we consider are those in which our query model: contains additional primitives not present in the reference; is missing primitives that are present in the reference. We will also look at cases in which there are a large number of primitives per model. These are all common issues facing any SLAM-type (simultaneous localisation and mapping) systems. To overcome these problems we introduce an *inter-object rotor magnitude-based matching function* and a *subsampling iterative rotor estimation and matching algorithm*. We title the finished algorithm: Rotor Estimation From Object Resampling and Matching—REFORM. REFORM builds on ideas from the RANSAC (RANdom SAMple Consensus) [7] and ICP (Iterative Closest Point) [3, 11] algorithms and extends these to multi-vector correspondence. It is easily parallelisable and designed for good convergence performance with models of real objects.

1. Introduction

A fundamental problem in computer vision is the correspondence problem. How do we match features from one image to another? This correspondence problem also appears when dealing with 3D data; given a reference model of an object and a query model of the same object how do we match objects, identify discrepancies and extract the transformation between the models? Our reference might be, for example, a CAD model, and our query model might represent the output of fitting primitives to LIDAR data or structure-from-motion point clouds. Many authors have tackled the problem of rotor estimation between groups of pre-matched geometric objects [5, 6, 12, 13] and

This article is part of the Topical Collection on Proceedings of AGACSE 2018, IMECCU-NICAMP, Campinas, Brazil, edited by Sebastião Xambó-Descamps and Carlile Lavor.

*Corresponding author.

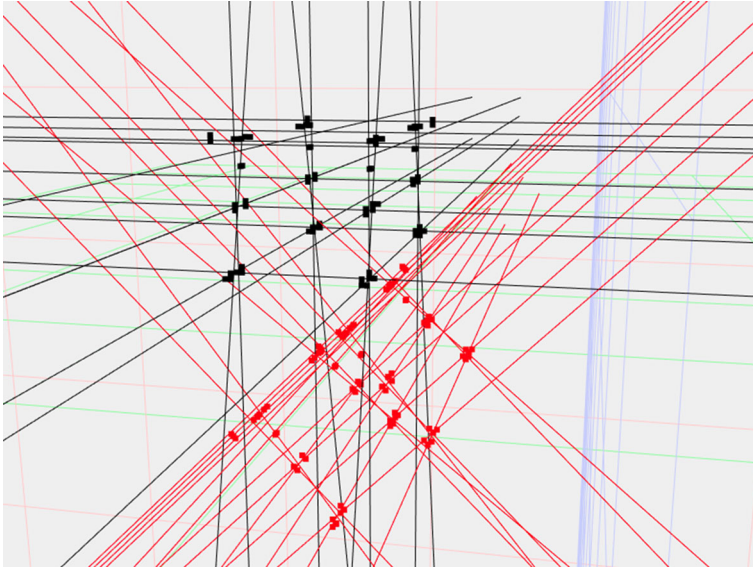


FIGURE 1. Black: 22 lines extracted from a CAD model of a table. Red: a transformation of the original model

others have applied conformal geometric algebra to 3D registration of point and sphere clouds [2,9]. In this paper we tackle the problem of registration and rotor estimation for primitives of any grade.

The objects we work with here will be CGA objects unless explicitly stated otherwise. We will use the standard extension of the 3D geometric algebra, where our 5D CGA space is made up of the standard spatial basis vectors $\{e_i\}$ $i = 1, 2, 3$, plus two additional basis vectors, e and \bar{e} with signatures, $e^2 = 1$, $\bar{e}^2 = -1$. Two *null vectors* can therefore be defined as: $n_\infty = e + \bar{e}$ and $n_0 = \frac{e - \bar{e}}{2}$. The mapping of a 3D vector x to its conformal representation X is given by $X = F(x) = \frac{1}{2}(x^2 n_\infty + 2x - 2n_0)$.

2. Proximity-Based Matching

Our first attempt at matching models made from a collection of geometric objects comes simply from considering their locality in space. For cases in which our query model is a small displacement (where *displacement* here will refer to rotation and translation) from the reference model, we would expect that simply assigning each object in the query model to its *closest* object in the reference model would give us a good number of correct matches.

Several authors have proposed cost functions between objects [12,13], and while many of these are extremely effective for extracting motors between circles and other round elements, they tend to fail to extract the transformation between parallel lines and planes. To counteract this problem we choose the cost function described in [6] (the properties of this cost function are further explored in [6]).

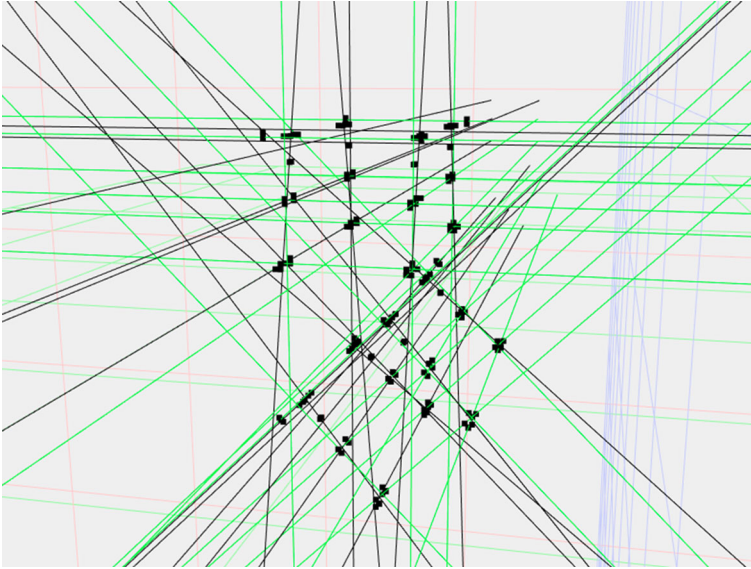


FIGURE 2. Using a direct proximity match between objects in the example scene, the green lines are correctly matched, the black lines are incorrectly matched. In this case the method produces 11 out of 22 correct matches

Consider first two arbitrary objects in 3D space represented as O_i and O_j in our conformal model. As in [10] we will extract the rotor R_{ij} that takes one object O_i to another O_j . Note that the objects will have an orientation (sign), and the rotor extraction will be orientation dependent. Once we have our rotor R between our conformal objects, the next step is to use this rotor to define a cost C as a function of this rotor:

$$C(R) = \langle (R - 1)(\tilde{R} - 1) \rangle_0 + \langle (R \cdot e)(R \cdot e)^\sim \rangle_0 \quad (1)$$

where $\langle X \rangle_r$ indicates the r -grade part of X . Equipped with this idea of closeness of objects, for a given i , a query object O_i is assigned to each of the reference objects O_j (i.e. this is done for all j), assuming the model and query sets are spatially close. For each object pair we form the rotor, R_{ij} that takes the query object to the reference object. The minimum cost assignment is then taken as the correct match, M_i , for that query object

$$M_i = \arg \min_j [C(R_{ij})]$$

Repeating this for all i , we define the total cost of this specific matching by summing the costs of each object-to-object match

$$C_{\text{total}} = \sum_i C(R_{iM_i})$$

The lower this cost, the better the models are matched. Figure 1 shows an example scene constructed of two line-based models extracted from a CAD

drawing, one model is in black and the other in red, the vertices of the models are also shown but are not used for matching. Figure 2 shows the result of performing proximity matching on the models, the lines in green are correctly matched and those in black are incorrectly matched. In this scene 11 of 22 lines are correctly matched by proximity matching.

3. Finding the Rotor Between Two Sets of Matched Objects

Given a set of matches for all object-pairs (under the assumption that the matching is correct) we need a method for finding the rotor between the two sets of objects. One technique for doing this is to optimise over our possible rotors, via minimisation of a cost function. Typically in CGA we parameterise and optimise over rotors in bivector space. Using the above cost metric it is shown in [6] that given correct matching we are able to perform non-linear convex optimisation and produce the correct rotation and displacement rotor. The downside of estimated gradient non-linear convex optimisation methods is that they typically require many cost function evaluations to reach the minimum, and when we have large numbers of objects in each model the optimisation can be very slow.

Here we propose an alternative Algorithm 1, based on directly using the rotors that we calculate between matched objects as part of the proximity matching procedure:

Algorithm 1: Direct rotor estimation algorithm

```

Result:  $R_e$ 
 $R_e = 1$  // The running estimate of the rotor;
for  $j \leftarrow 0$  to  $max\ iterations$  do
     $R_s = 1$  // Keep track of the rotor as we iterate over all the
    objects;
    for  $m \leftarrow 0$  to  $N\ matches$  do
         $U_m = R_e Q_m \tilde{R}_e$  // Transform query object  $Q_m$  by the
        current rotor  $R_e$ ;
         $R_m = \text{rotor between objects}(U_m, O_m)$  // R from transformed
        object  $U_m$  to matched ref object  $O_m$ ;
         $R_r = \sqrt{R_m}$  // Take the square root of the match rotor;
         $R_e = R_r R_e$  // Update the running estimate to use  $R_r$ ;
         $R_s = R_r R_s$  // Update the rotor for this iteration;
    end
    if  $R_s = 1$  then
        return  $R_e$  // Terminate when the full loop over the objects
        comes back on itself;
    end
end

```

This algorithm does not require the computing of an explicit cost function, it is heuristic driven and has not been proven to converge. In practice

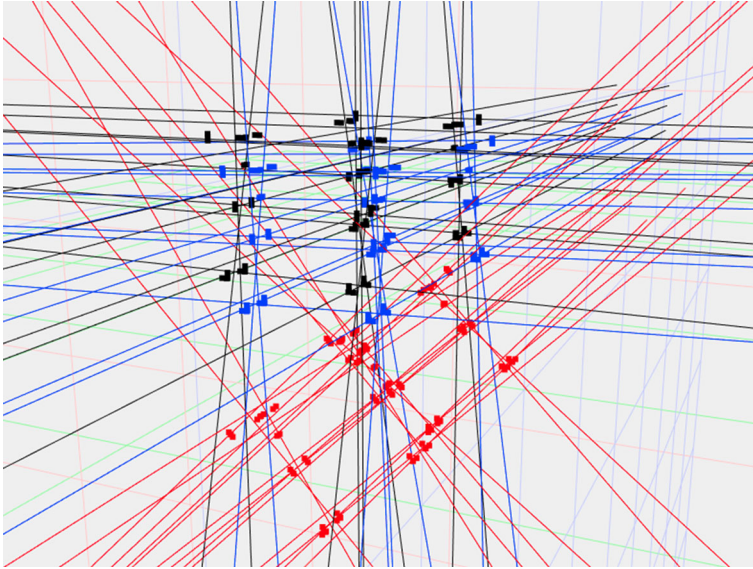


FIGURE 3. The blue model is the estimated transformation from the set of red lines to the set of black lines (see Fig. 1) given the starting proximity match using the non linear optimisation method. Given an initial proximity matching (see Fig. 2), the rotor found by non-linear optimisation still puts the models in close proximity even if the initial matching is not perfect

however we have found it to perform well. In the case of a fully correct matching, the rotor that is found, for both the non-linear optimisation algorithm and this direct algorithm, is indeed the rotor that takes our query model to our reference model. In the case of a partially incorrect initial matching, the rotor that is produced typically takes the query model closer to the reference model but does not produce the true rotor as shown in Figs. 3 and 4.

4. Iterative Matching and Rotor Estimation

Armed with rotor estimation techniques for correctly matched reference and query models we will move to more difficult situations. Consider the general case where the query and reference models are not in close proximity. In this situation we first make an initial guess at the object matches and estimate the rotor between the query and reference models using the methods described in the previous section. If our initial matching was not completely correct we will not estimate the correct rotor between the objects, the resultant rotor will have some error but will likely be relatively close to the true rotor. If we transform our query model by the estimated rotor we can use proximity matching between the transformed query model and the reference to get a better set of object matches. The process is then repeated so that the number

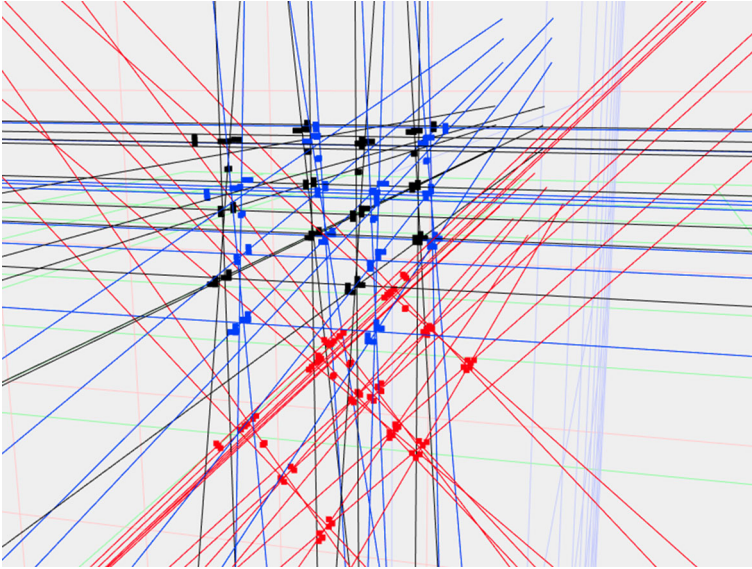


FIGURE 4. As in Figure 3, the blue model is the estimated transformation from the red to the black given the starting proximity match, in this case using the direct rotor estimation algorithm. The direct rotor estimation algorithm in practice produces rotors of similar quality to the non-linear optimiser

of incorrect matches decreases with each iteration and the process converges. The iterative algorithm is summarised in the following:

1. Each object in the query model is given a match in the reference model (there are a number of ways of making this initial guess)
2. Calculate the rotor between the models assuming the current matches are correct, this can be done by running an optimisation algorithm to completion or by using the direct method mentioned in the previous section.
3. Transform the query model by applying the rotor calculated in the previous step
4. Each object in the transformed query model is compared to each object in the reference model, the match with the minimum cost according to our chosen cost function is accepted
5. If there is no change in the matches terminate the algorithm otherwise go back to step (2)

This algorithm correctly handles partially incorrect initial matching between models, and iterates towards the answer in relatively few steps. It is also deterministic, each step is a function only of the current state and it has fixed termination criteria that clearly indicate when it has completed. In its

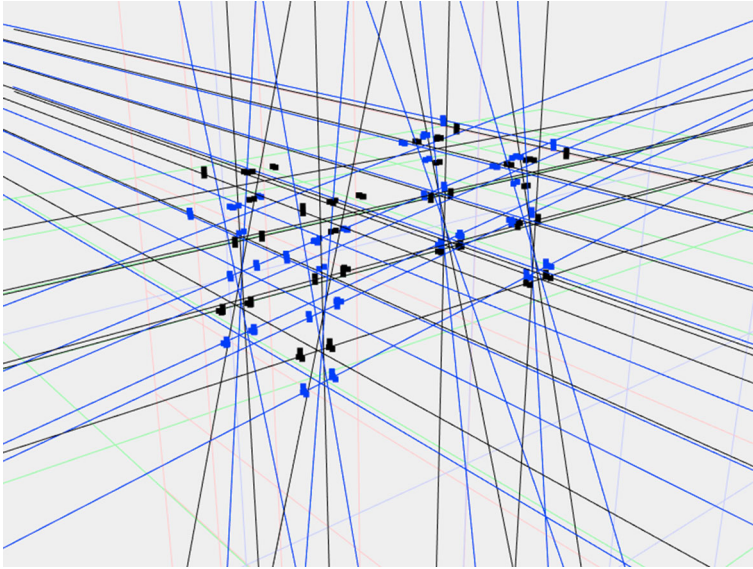


FIGURE 5. As our model contains a lot of symmetry the iterative algorithm is prone to getting stuck in local minima. As in Fig. 1 the black model is our reference and the blue is our estimate, the red lines are not shown for clarity. Here the blue model is at the final output of the iterative matching algorithm. 17 of 22 matches are correct but the algorithm is stuck in a local minima

current state this algorithm is an extension of the well known iterative closest point (ICP) algorithm [3, 11] routinely used for point cloud registration. As with the ICP algorithm, a significant problem arises when we consider cases in which large fractions of the initial matches are incorrect, resulting in convergence to an incorrect set of correspondences. With our geometric algorithm we additionally see local minima arise when models contain many parallel lines or planes and computationally we run into trouble when models contain a very large number of geometric objects. In these cases the algorithm may fail to converge to the true rotor and instead become stuck in a local minimum even though some matches are correct. Real manufactured objects or buildings typically contain many parallel faces and lines and as such we need a way to overcome these limitations. Figure 5 shows an example of the previously studied scene stuck in a local minima, in this case there are 17 of 22 lines correctly matched but the algorithm will not progress further.

5. Incorporating Sampling

To counteract the local minima issue, we modify our procedure to incorporate sampling in a RANSAC-like [7] algorithm. This particular approach is chosen as it is readily adapted to parallel processing and is well suited to handling

large numbers of incorrect matches. After each matching stage in the previous algorithm we randomly and uniformly sample m lots of k matches. Each of these m match sets then propagates through the rotor estimation algorithm and each produces a candidate rotor for the model matching and a cost associated with that rotor for these k matches. The rotor produced by the sample with the minimum cost is then chosen and used to transform the entire query model. This repeats for a fixed number of iterations or until some cost threshold is reached.

The full REFORM algorithm is now summarised as follows:

1. Each object in the query model is given a match in the reference model (there are a number of ways of making this initial guess)
2. Given our matches, randomly select multiple sample subsets
3. For a given sampled subset calculate the rotor that leads to minimum total cost between the subset objects as in Eq. (1)
4. Accept the rotor from the sample that gives the minimum total cost between the subset objects
5. Update our query model position by applying the estimated rotor
6. Each object in the query model is compared to each object in the reference model, the match with the minimum cost is accepted
7. Check termination criteria, go back to step 2.

The disadvantage of moving to a sampling-based model is that we no longer have fixed termination criteria—just because the matches have not changed over multiple sampling and optimisation steps, does not mean they will not change as a result of the next one. On the other hand, the rotor estimation and cost calculation for each sample is independent of every other sample allowing for easy parallelisation. The subsampling also allows the algorithm to jump out of local minima by sampling correct matches whose effect would normally be swamped by the mass of incorrect matches. A CUDA implementation of the algorithm has been written, leveraging the massive parallelisation capability afforded by modern graphics cards and is incorporated in the clifford python package [1].

6. Matching Scenes of Mixed Geometric Primitives

3D models of objects are typically constructed from collections of geometric objects, planes, lines and points. While traditional matching techniques typically use points from meshes [8] or points derived from the intersection of planes/lines [4], REFORM allows us to incorporate multiple types of 3D object together into the same matching and rotation/translation estimation framework, Fig. 6 shows an example of two matching synthetic models composed of both lines and planes, in this example REFORM handles both types of object transparently.

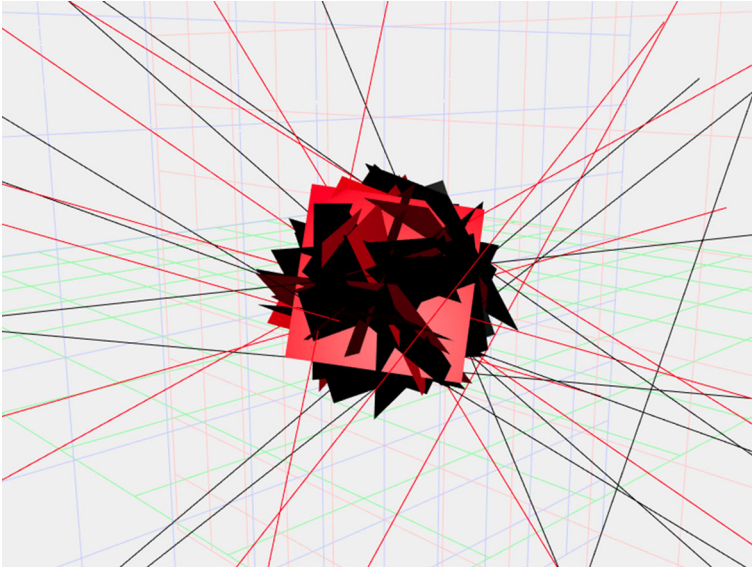


FIGURE 6. Sets of synthetic random lines and planes in red along with their transformation in black to be matched. REFORM handles both in the same framework and correctly extracts the rotor between them

7. Conclusions

In this paper we have presented an algorithm for registering models composed of geometric primitives. This algorithm extends the range of traditional matching and registration algorithms from point cloud only techniques to incorporate higher grade geometric objects. The solution is available in the clifford [1] python package with both CPU and GPU implementations.

Open Access. This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

- [1] Arsenovic, A. Hadfield, H., Kern, R.: The Pygae Team. pygae/clifford: v1.0.1 (2018)
- [2] Bayro-Corrochano, E., Rivera-Rovelo, J.: The use of geometric algebra for 3d modeling and registration of medical data. *J. Math. Imaging Vis.* **34**(1), 48–60 (2009)

- [3] Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **14**(2), 239–256 (1992)
- [4] Bosche, F.: Plane-based coarse registration of 3d point clouds with 4d models. In: *28th International Symposium on Automation and Robotics in Construction* (2011)
- [5] De Keninck, S., Dorst, L.: Geometric algebra Levenberg–Marquardt. Pre-print, private communication (2019)
- [6] Eide, E.R., Lasenby, J.: A novel way of estimating rotors between conformal objects and its applications in computer vision. *AACA: Topical Collection AGACSE 2018, IMECC-UNICAM, Campinas, Brazil* (2018)
- [7] Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**(6), 381–395 (1981)
- [8] Gelfand, N., Mitra, N.J., Guibas, L.J., Pottmann, H.: Robust global registration. *Eurographics Symposium on Geometry Processing*, page 10 (2005)
- [9] Kleppe, A.L., Egeland, O.: A curvature-based descriptor for point cloud alignment using conformal geometric algebra. *Adv. Appl. Clifford Algebras* **28**(2), 50 (2018)
- [10] Lasenby, J., Hadfield, H.: Calculating the rotor between conformal objects. *AACA: topical collection AGACSE 2018, IMECC-UNICAM, Campinas, Brazil* (2018)
- [11] Segal, A.V., Haehnel, D., Thrun, S.: *Generalized-icp*. *Robotics: Science and Systems* (2009)
- [12] Tingelstad, L., Egeland, O.: Motor estimation using heterogeneous sets of objects in conformal geometric algebra. *Adv. Appl. Clifford Algebras* **27**(3), 2035–2049 (2017)
- [13] Valkenburg, R., Dorst, L.: Estimating motors from a variety of geometric data in 3d conformal geometric algebra. *Journal of Theoretical Biology*, pp. 25–45 (2011)

H. Hadfield and J. Lasenby

Signal Processing and Communications Group, Department of Engineering
University of Cambridge
Cambridge
UK

e-mail: hh409@cam.ac.uk

J. Lasenby

e-mail: j1221@cam.ac.uk

M. Ramage

Department of Architecture
University of Cambridge
Cambridge
UK

e-mail: mhr29@cam.ac.uk

C. Doran
Sidney Sussex College
University of Cambridge
Cambridge
UK
e-mail: cjld1@cam.ac.uk

Received: February 28, 2019.

Accepted: June 13, 2019.