Target Calibration and Tracking Using Conformal Geometric Algebra

Yilan Zhao[1], Robert Valkenburg[2], Reinhard Klette[1], and Bodo Rosenhahn[3]

[1] Computer Science Department, The University of Auckland, New Zealand
[2] Industrial Research Limited, Auckland, New Zealand
[3] Max Planck Institute, Saarbrücken, Germany

Abstract. This paper is about real-time refinement of the 3D positions of a large number of stationary point-targets from a sequence of 2D images which are taken by a hand-held, calibrated camera group. To cope with the large data quantity arriving rapidly, an efficient iterative algorithm was developed. The problem and solution are expressed entirely within the computational framework of conformal geometric algebra. The iterative solution requires a pose estimation step of which two strategies are investigated. Experiments are performed to evaluate the algorithm based on synthetic and real data.

Keywords: conformal geometric algebra, pose estimation.

1 Introduction

Recovering the positions of many point-targets over a large area is computationally expensive. This paper describes an efficient iterative algorithm to refine target positions from a sequence of 2D images. The targets used in this project are point-lights (left Figure 1) and form part of a flexible 6D positioning system. A group of rigidly co-located calibrated cameras (right Figure 1) is moved along an arbitrary path and takes images of the targets. The image points of the targets are transformed to 3D lines which are used by the algorithm to update the 3D positions of the targets. The algorithm is expressed entirely within the computational framework of conformal geometric algebra (CGA). The previously developed target calibration algorithm described in [9] is non-iterative and requires all the line data to be gathered before the algorithm can proceed. It can be used to obtain an initial estimate of the target positions for the iterative algorithm described in this paper. This work is a continuation of work reported in [9] in the application of the conformal model of geometric algebra.

1.1 Geometric Algebra and Conformal Model

In this section, the basic concepts and operations of geometric algebra that are required in this paper are briefly introduced. For a detailed introduction to geometric algebra, refer elsewhere e.g. [1,2,3].

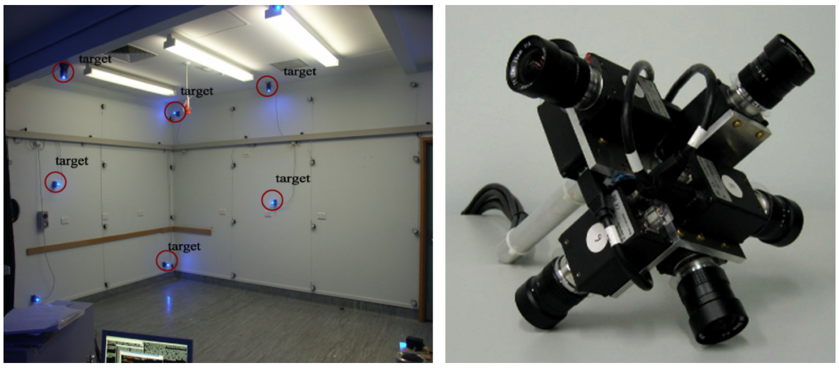

Fig. 1. Left: targets; six of them are encircled. Right: camera group.

Geometric algebra (GA) is the application of Clifford algebras to geometric problems. It integrates many concepts and techniques, such as linear algebra, vector calculus, differential geometry, complex numbers and quaternions, into a coherent framework. A geometric algebra over $\mathbb{R}$ is denoted $\mathcal{G}_{p,q}$ with p positive and q negative basis elements. Let x_1, x_2, …, x_r be vectors. $X = x_1 \wedge x_2 \wedge \ldots \wedge x_r$ is referred to as an r-blade where '$\wedge$' is called *outer product*. r is the *grade* which indicates the dimensionality of the blade. A linear combination of multiple r-blades constructs an r-*vector*. $\mathcal{G}^r_{p,q}$ denotes the r-vectors in $\mathcal{G}_{p,q}$. A linear combination of a set of elements with different grades is a *multivector*. For example, if A is a multivector then it can be written as $A = \sum_r \langle A \rangle_r$ where $\langle A \rangle_r$ represents the grade r part of A. $\langle A \rangle$ or $\langle A \rangle_0$ represents the scalar part of A. The part of A containing the grades in another multivector B is denoted as $\langle A \rangle_B$. $A \rfloor B = \Sigma_{r,s} \langle \langle A \rangle_r \langle B \rangle_s \rangle_{s-r}$ is defined as the left contract inner product of A and B. The outer product can be related with the inner product by the following equation: $A \rfloor (B \rfloor C) = (A \wedge B) \rfloor C$. *Reverse* of X is defined as $\widetilde{X} = x_r \wedge \ldots \wedge x_2 \wedge x_1$. The dual of a blade X is defined as $X^* = X \rfloor I^{-1}$, where the pseudo-scalar I is an n-blade $e_1 \wedge \ldots \wedge e_n$ based the orthogonal basis ($\{e_i : \ i = 1 \ldots n\}$, $e_i \cdot e_j = 0$ for $i \neq j$, $e_i \cdot e_i = 1$) of $\mathbb{R}^n$ within $\mathcal{G}_n$. The norm of a multivector A can be calculated by $|A| = \sqrt{\left| \left\langle \widetilde{A} A \right\rangle \right|}$. If S is a linear operator, the *outermorphism* $\underline{S}$ is defined by $\underline{S}(X) = S(x_1) \wedge S(x_2) \ldots \wedge S(x_r)$. The derivative of multivector valued function F with respect to multivector X is denoted $\partial_X F$. $\dot{\partial}_X F \dot{G}$ means differentiate $G = G(X)$ with respect to X while regarding F as a constant. The following result [10] is required in later developments,

$$\partial_X \left\langle XYX^{-1}Z \right\rangle = \left\langle YX^{-1}Z \right\rangle_X - \left\langle X^{-1}ZXYX^{-1} \right\rangle_X$$

where X, Y, Z be multivectors where Y and Z are independent of X.

GA expresses a number of models of 3D Euclidean space ($\mathcal{E}^3$), such as 3D Euclidean model, 4D homogeneous model and 5D conformal model. In this paper we use the conformal model of geometric algebra (CGA) based on $\mathcal{G}_{4,1}$. $\mathcal{G}_{4,1}$ is based on the orthonormal basis $\{e_1, e_2, e_3, e_+, e_-\}$ where $e_k^2 = e_+^2 = 1$ and $e_-^2 = -1$. It is usually more convenient to use the basis $\{e_o, e_1, e_2, e_3, e\}$ as it has a better geometric interpretation, where $e_o = \frac{e_- - e_+}{2}$ is associated with the origin and $e = e_- + e_+$ with the point at infinity. CGA allows a diversity of objects to be represented directly as blades (e.g. point, line, plane, circle, sphere, tangents and orientations) and allows a variety of operations to be represented as versors (e.g. rotor, translator, motor). A vector is represented as $v = v_1e_1 + v_2e_2 + v_3e_3$ where v_1, v_2, v_3 are scalars. A point with location at the Euclidean point $\boldsymbol{p} \in \mathcal{G}_3^1$ is represented as $p = \boldsymbol{p} + e_o + \frac{1}{2}\boldsymbol{p}^2 e \in \mathcal{G}_{4,1}^1$. A line is represented by $\Lambda = p \wedge v \wedge e$ where $p \in \mathcal{G}_{4,1}^1$ is a point and $v \in \mathcal{G}_3^1$ is a direction vector. A line is normalised by the mapping $\Lambda \to \frac{\Lambda}{\|\Lambda\|}$. A dual sphere centered at point p with radius ρ is given by $s = p - \frac{1}{2}\rho^2 e$. A Euclidean motion is represented by a *motor* $M = \exp\left(-\frac{1}{2}B\right)$ where $B = \boldsymbol{B} - \boldsymbol{t}e$ where $\boldsymbol{B} \in \mathcal{G}_3^2$ and $\boldsymbol{t} \in \mathcal{G}_3^1$. A motor M has properties which are important for deriving the algorithm: (i) $M \in \mathcal{G}_{4,1}^{0,2,4}$, (ii) $M\widetilde{M} = 1$, (iv) if $X \in \mathcal{G}_{4,1}^k$ then the transformation of X is given by $MX\widetilde{M} \in \mathcal{G}_{4,1}^k$.

1.2 Problem Description

The targets are defined in a world coordinate system denoted by CSW. Since the geometric relationship between the individual cameras which comprise the camera group is fixed and known, the camera group can be associated with a single moving coordinate system denoted by CSM.

An initial estimate of the positions of n targets $\{p_i^0 \in \mathcal{G}_{4,1}^1, i = 1 \ldots n\}$ is given [9]. The initial pose of the camera group CSM is also given and represented as a motor M_o. The camera group CSM is moved to m positions on the path in CSW. The movement of CSM is tracked and represented by a sequence of motors $M_k, k = 1 \ldots m$. At each position in CSW, a set of images are captured and the image points of the targets are extracted and converted to normalised lines $\{\Lambda_i^k \in \mathcal{G}_{4,1}^3, i = 1 \ldots n, k = 1 \ldots m\}$ in CSM. These lines are processed to refine the initial target position estimates. When CSM is moved to the next position, the new estimate of target positions will be calculated based on the previous estimate and a new set of lines. For m positions on the path, m updates are performed.

The problem can now be summarised as follows: Given a group of lines in CSM, a previous estimate of a set of points and a previous pose, we wish to update the coordinates of these points in CSW.

2 Target Refinement Using Geometric Algebra

The solution to the problem is analysed and developed in this section. At the beginning of the motion of the camera group we are given initial positions of targets and the initial pose of CSM. At each position we are given a new set of lines between optical centers and visible targets in CSM. The following steps need to be done during camera motion: (i) pose estimation of CSM; (ii) transformation of corresponding lines from CSM into CSW; (iii) update of target positions.

2.1 Pose Estimation: Objective Function Versus Point-Line Constraint

We estimate the pose of CSM by two alternative iterative strategies (i) non-linear optimisation of an objective "error" function. (ii) root finding of a 4-blade *point-line constraint* equation.

Non-linear optimisation of an objective function. The distance d between a point p and a line Λ is defined [10] by $d^2(p, \Lambda) = -\frac{1}{2}\langle \Lambda p \Lambda p \rangle$. The total distance between all points and their associated lines is defined as follows:

$$d^2 = \sum_j \sum_i \alpha_i \left(d^2(p_i, \Lambda_j) \right) \tag{1}$$

where $\alpha_i \in \{0, 1\}$ indicates whether the target is visible by any of the cameras. p_i is a target point and Λ_j is assumed to be a line which connects p_i to different cameras (i.e., their optical centers) in CSW. If the lines are given in CSM and the pose of CSM is represented by M then Λ in Equation (1) is replaced by $M\Lambda\widetilde{M}$ giving

$$d^2(M) = -\frac{1}{2} \sum_i \sum_j \alpha_i \left\langle (M\Lambda_j\widetilde{M}) p_i (M\Lambda_j\widetilde{M}) p_i \right\rangle \tag{2}$$

This objective function produces a scalar with a well-defined geometric meaning.

The poses of CSM are estimated using a Quasi-Newton optimization technique which is described in [6] (pages 425–430). We use a non-linear minimisation routine (called "dfpmin") which implements the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) update.

The optimization routine requires an objective function and its gradient. The motor M representing the pose of CSM is parameterised $M = M(x)$ where $x \in \mathbb{R}^6$. We use $M(x)$ in the objective function d^2 in Equation (2) to express the objective function as $g(x) = d^2(M(x))$. The gradient is given by $[\nabla_x g(x))]_i = \partial_{x_i} g(x) = \partial_{x_i} M * \partial_M d^2$. The derivative $\partial_M d^2$ is calculated as follows:

$$\begin{aligned} \partial_M d^2 &= -\frac{1}{2} \partial_M \left\langle M\Lambda\widetilde{M} p M\Lambda\widetilde{M} p \right\rangle \\ &= -\left\langle \Lambda\widetilde{M} p M\Lambda\widetilde{M} p \right\rangle_M + \left\langle \widetilde{M} p M\Lambda\widetilde{M} p M\Lambda\widetilde{M} \right\rangle_M \end{aligned} \tag{3}$$

where M must be a motor so $\widetilde{M} = M^{-1}$. The operator $\langle \ldots \rangle_M$ denotes the projection of a general multivector onto the grades being present in multivector M. The optimisation returns the estimated parameters x of the motor $M(x)$.

Root finding of a point-line constraint equation. An alternative distance measure is expressed in an implicit way by the equation

$$p \wedge (M \mathit{\Lambda} \widetilde{M}) = 0 \tag{4}$$

which indicates point p is on line $\mathit{\Lambda}$. We call this the *point-line constraint.*

For all target points, the point-line constraint becomes

$$\sum_i \alpha_i \left(\sum_j p_i \wedge (M \mathit{\Lambda}_j \widetilde{M}) \right) = 0 \tag{5}$$

where $\alpha_i \in \{0, 1\}$ indicates whether the target is visible by any of the cameras. This point-line constraint expresses a geometric distance measure and is commonly applied in computer vision, see [5,8].

This technique uses the point-line constraint in Equation (4) for distance measurement. Given the previous motor M_{k-1}, M_k can be estimated as $M_k = \mathit{\Delta} M_k M_{k-1}$. Assume the previous pose M and line $\mathit{\Lambda}$ are known. Let us update the current pose $\mathit{\Delta} M M$. The constraint becomes

$$\left(\widetilde{\mathit{\Delta} M} p' \mathit{\Delta} M \right) \wedge \mathit{\Lambda} = 0 \tag{6}$$

where $p' = \widetilde{M} p M$ represents a point in the previous CSM. $\mathit{\Delta} M$ needs to be estimated.

In order to solve for $\mathit{\Delta} M$, it is necessary to linearise the motor part (i.e., $\widetilde{\mathit{\Delta} M} p' \mathit{\Delta} M$) of the equation. The motion of the camera group is considered as a general motion, which is formulated using an exponentiated bivector (2-vector); $\mathit{\Delta} M$ is expressed in the form

$$\exp\left(-\frac{\mathit{\Delta} B - \mathit{\Delta} t e}{2} \right)$$

where $\mathit{\Delta} B$ is a Euclidean bivector and $\mathit{\Delta} t$ is a vector.

The Euclidean transformation (i.e., $\mathit{\Delta} M$) of a point p' can be approximated as follows:

$$\begin{aligned} \widetilde{\mathit{\Delta} M} p' \mathit{\Delta} M &= \exp\left(\frac{\mathit{\Delta} B - \mathit{\Delta} t e}{2} \right) p' \exp\left(-\frac{\mathit{\Delta} B - \mathit{\Delta} t e}{2} \right) \\ &\approx \left(1 + \frac{\mathit{\Delta} B - \mathit{\Delta} t e}{2} \right) p' \left(1 - \frac{\mathit{\Delta} B - \mathit{\Delta} t e}{2} \right) \\ &\approx p' - p' \rfloor \mathit{\Delta} B + p' \rfloor (\mathit{\Delta} t e) \end{aligned} \tag{7}$$

In Equation (7), two approximations are involved. The first approximation involves truncating the Taylor series for $\exp(X)$ (i.e., $\exp(X) \approx 1 + X + \frac{X^2}{2!} + \cdots$). The second approximation involves removing second order terms from the final product; this works well only when the motion $\mathit{\Delta} M$ is sufficiently small (say, its rotation angle is smaller than 10 degree). This condition is satisfied when the camera group moves “smoothly” along its path and is sampled sufficiently frequently.

A similar linearisation of a transformation for a single point using different expressions is described in [8]. By substituting the approximated expression of $\widetilde{\Delta M} p' \Delta M$ given by Equation (7) back into constraint Equation (6), the constraint becomes

$$p' \wedge \Lambda - (p' \rfloor \Delta B) \wedge \Lambda + (p' \rfloor (\Delta t e)) \wedge \Lambda = 0 \tag{8}$$

with two unknowns: ΔB and Δt. Therefore, ΔM is calculated by estimating ΔB and Δt. A set of point-line correspondences are required to solve for ΔB and Δt in Equation (8). As any linear geometric algebra equation can be expressed in matrix form, we solve the equation by solving the associated matrix system of the form $Ax = b$. This can be solved by any standard technique such as LU decomposition. From x we obtain ΔB and Δt and hence ΔM. Each calculated ΔM provides a step towards the desired motor and this process is repeated until convergence. The first step towards the target motor is denoted by ΔM_1. By repeating this procedure, $M_{k2}, \ldots, M_{kn}$ are estimated, which converge towards M_k where n iterations are necessary. ΔM is calculated as $\Delta M_n \ldots \Delta M_2 \Delta M_1$. The convergence rate depends on the “speed” of the expected transformation (i.e., the movement of the cameras within the space where images are taken). We stop the approximation (iteration) if $\|\Delta M_i\| \leq \epsilon$ (e.g., $\epsilon = 10^{-6}$), which indicates that no further improvement can be achieved. Several iterations are usually sufficient to obtain the next pose of the camera group.

2.2 Update Target Positions

With the estimated pose M of CSM, the given lines Λ in CSM can be transformed to CSW by $M\Lambda\widetilde{M}$. Given all the lines in CSW for all poses, the current target positions can be calculated by Lemma 1 [9],

Lemma 1. *Let $\Lambda_j \in \mathcal{G}^3_{4,1}$, $j \in J$ be a set of normalised lines and $S(x) = \sum_{j \in J} S(x, \Lambda_j)$ where $S(x, \Lambda_j) = x - (x \rfloor \Lambda_j) \rfloor \Lambda_j$. If $\underline{S} I_3 \neq 0$ then the point $q \in \mathcal{G}^1_{4,1}$ closest to all the lines in the least squares sense is given by the center of the normalised dual sphere*

$$s = -\frac{\underline{S}(I_3) \rfloor I_4}{\underline{S}(I_3) \rfloor I_3} \tag{9}$$

where $I_3 = e_1 \wedge e_2 \wedge e_3$ and $I_4 = e_o \wedge e_1 \wedge e_2 \wedge e_3$.

As the target positions are estimated in real time, an increasingly large number of lines and frequently repeated calculations would require too much computational resource. Rather than storing all the lines we update some summary variables to implement an iterative algorithm.

In Lemma 1, $\underline{S}(I_3)$ and $\underline{S}(I_4)$ depend on all lines and vary with each update. As $\underline{S}(I_3) = S(e_1) \wedge S(e_2) \wedge S(e_3)$ and $\underline{S}(I_4) = S(e_o) \wedge S(e_1) \wedge S(e_2) \wedge S(e_3)$ it is only necessary to store and update $S(e_o)$, $S(e_1)$, $S(e_2)$ and $S(e_3)$. During the iterations, the information contained in the lines needed for estimating the target positions, are accumulated in $S(e_o)$, $S(e_1)$, $S(e_2)$ and $S(e_3)$. Recall S is defined as $S(q) = \sum_{i=1}^{n}(q - (q \rfloor \Lambda_i) \rfloor \Lambda_i)$. The current estimate of $S(e_j)$ can be updated based on previous $S_{k-1}(e_j)$, and new lines $\Lambda_i, i \in I_k$ arriving at current time k as

$$S_k(e_j) = S_{k-1}(e_j) + \sum_{i \in I_k}(e_j - (e_j \rfloor \Lambda_i) \rfloor \Lambda_i) \quad (10)$$

It is not necessary to update the targets on every pose update iteration. For example, the targets may be updated after CSM has been moved by some specified distance.

3 Experiments

Experiments were carried out using both simulated data and real data. Both kinds of data allowed us to test the validity and performance of our algorithm using both the point-line constraint and the objective function (Quasi-Newton optimisation) pose update. Noise was added to test the stability of the algorithm.

3.1 Simulated Data

In order to test and evaluate the iterative algorithm for estimating target positions, we generated simulated line data. We have the ground truth target position obtained using a total station. We generated a synthetic path for CSM in a real scene (a lab at Industrial Research Ltd.). Synthetic lines were created using this path and projecting the known targets through the real calibrated camera group model. In order to test the behaviour of the algorithm in the presence of noise we generated simulated data with different levels of noise. The stability of the algorithm is investigated by adding Gaussian noise with deviation $\sigma \in [0.2, 1.0]$ pixels (see Figure 2).

With the minimum noise, the errors of estimation decrease smoothly by around 30%. With more noise, the error curve fluctuates within a wider range. But the error is still reduced as the update process continues. Even with the maximum noise, the target position is refined by around 20%. We applied the simulated data to both algorithms. Both algorithms are validated by a comparison of experimental results with ground truth, and also between both. Table 1 shows comparisons for estimating different poses of CSM along a 3D path.

Comparisons showed that both pose update strategies achieve almost the same results. The strategy using the point-line constraint was nearly twice as fast as Quasi-Newton strategy. This can be partly attributed to the fact that the point-line constraint method make no effort to guarantee global convergence. The Quasi-Newton method proved more robust under all considered conditions, and the point-line constraint method is limited to the condition that differences between subsequent poses are small because no global convergence protection was implemented.

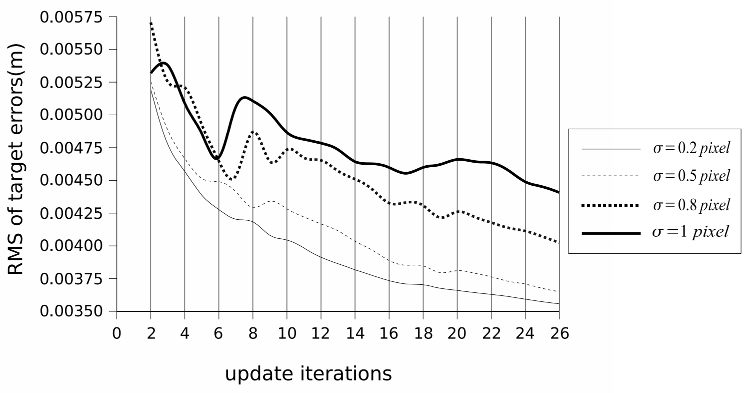

Fig. 2. The *RMS* (Root Mean Square) of errors in targets vs update iterations with different levels of noise

Table 1. Comparison of results for the two alternative pose estimation strategies for the kth pose (rotation and translation). θ_1 and t_1 are rotation angle (in degree) and translation vector (in millimeter) of the pose using the quasi-Newton method; θ_2 and t_2 are those for the line-target constraint method.

k	$(\theta_1 - \theta_2) \times 10^{-4}$	$\|t_1 - t_2\| \times 10^{-4}$
1	1.23	1.44
5	2.11	0.84
10	0.67	2.10
15	1.01	1.25
20	0.19	0.09
26	3.61	0.56

3.2 Real Data

Real data sequences of images captured by the camera group are shown in Figure 1, right. The lab room is visualised using VRML software; see Figure 3. Results for real data were not as good (for both pose update strategies) as for simulated data.

We believe that this can be partially explained by small errors in the camera group model. A better camera group calibration should reduce these errors. During simulation the same camera group model is used for projection (targets mapped to image points) and backprojection (image points mapped to lines) so any calibration errors have no influence.

Estimated poses and target positions are also visualised in Figure 3. Comments about performance comparisons between both estimation methods apply qualitatively for real data the same way as for simulated data. The target update algorithm run run at 30Hz on a standard 3GHz PC using either of the pose update schemes.

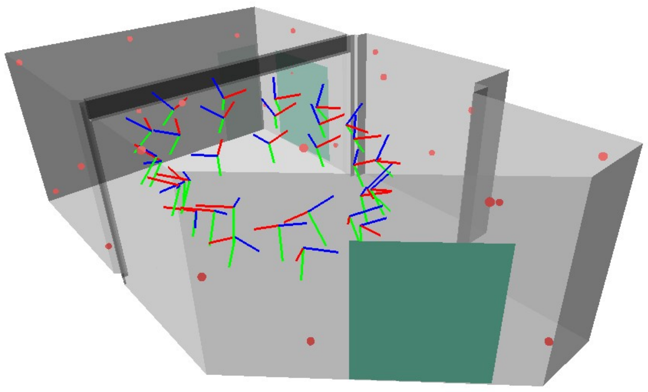

Fig. 3. A model of the lab space used. Disks are estimated targets; the figure also shows a few *CSM* coordinate systems along the path of the camera group.

4 Conclusion

We developed an iterative algorithm for refining 3D target positions over a large number of images. We acquire (from 2D images) lines pointing towards 3D targets. The use of the conformal model of geometric algebra (CGA) benefits the development of the solution in both theory and practice. CGA provides a compact symbolic representation of objects and their transformations. A variety of objects (e.g., vectors, points, lines, spheres) and operations (e.g. motors) can be represented in a single algebra which simplifies the implementation. The use of a single motor element to represent a Euclidean transformation (instead of separate rotation and translation), further simplified the implementation.

The iterative target update algorithm performed well over a wide variety of conditions. Two iterative strategies are used for pose estimation. The point-line constraint strategy proved to be more efficient than the Quasi-Newton optimisation strategy, but less robust in stability.

References

1. L. Dorst and S. Mann. Geometric algebra: a computational framework for geometrical applications, Part 1 and Part 2. *IEEE Computer Graphics Applications*, vol. 22, no. 3 and 4, 2002.
2. D. Hestenes, Old wine in new bottles: A new algebraic framework for computational geometry, In E. Bayro-Corrochano and G. Sobczyk, editors, *Geometric Algebra with Applications in Science and Engineering*, chapter 1, Birkhäuser, 2001.
3. T.F. Havel, Geometric algebra: parallel processing for the mind, In: *MIT Independent Activities Period Lectures*, 2002.
4. W. E. L. Grimson. *Object Recognition by Computer.* The MIT Press, Cambridge, MA, 1990.
5. G. A. Kramer. *Solving Geometric Constraint Systems: A Case Study in Kinematics.* ACM Distinguished Dissertations, MIT Press, 1992.

6. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C++*, 2nd Edition, Cambridge University Press, 2002.
7. B. Rosenhahn and G. Sommer. Pose estimation in conformal geometric algebra. Part I: The stratification of mathematical spaces. *J. Mathematical Imaging Vision*, **22**:27–48, 2005.
8. B. Rosenhahn and G. Sommer. Pose estimation in conformal geometric algebra. Part II: Real-time pose estimation using extended feature concepts. *J. Mathematical Imaging Vision*, **22**:49–70, 2005.
9. R. J. Valkenburg and N. S. Alwesh. Calibration of target positions using the conformal model and geometric algebra. In Proc. *Image Vision Computing New Zealand*, Otago University, pages 241-246, 2005.
10. R. J. Valkenburg, Some techniques in geometric algebra for computer vision, Tech. Rep. 87130001-1-03, Industrial Research Limited, August, 2003.
11. R. Wareham, J. Cameron, and J. Lasenby. Applications of conformal geometric algebra in computer vision and graphics. In Proc. *IWMM/GIAE*, pages 329–349, 2004.