Geometric Computing for Cybernetics

Eduardo Bayro Corrochano
edb@gdl.cinvestav.mx
http://www.gdl.cinvestav.mx/edb

CINVESTAV
Center of Research and Advanced Studies, Guadalajara
Jalisco, Mexico

Department of Electrical Engineering and Computer Science
GEOVIS Laboratory

MOIMA June.23..2016, Hanover
Content

- Motivation
- Mathematical Framework
- Robot Manipulators and Mobile Robots
- Humanoids Mechanisms
- Perception
- Medical Robotics
- Low and Middle Level Perception
- Conclusions
Motivation: building perception action systems

A sortiment of mathematical tools

Conformal geometric algebra
Modern unifying language for the design of algorithms for PAC systems
Motivation: shall we go for higher dimensions?

Are we sophisticated the approach for sake for mathematical elegance?

Do we gain something by using other kind of geometric representations?

Real Time?
Geometric Algebra
Evolution of Algebras

Quaternions, o a new system of imaginaries in algebra

Die lineale Ausdehnungslehre ein neuer Zweig der Mathematik

Clifford algebra: fusion of Grassman and Hamilton ideas

William Rowan Hamilton

Hermann Günther Grassmann

William Kingdon Clifford
Hongbo. Li, David Hestenes and Alyn Rockwood. "Generalized Homogeneous coordinates for computational geometry".
Geometric Product of two vectors

The geometric product of two vectors \( \mathbf{a} \) and \( \mathbf{b} \) is written as \( \mathbf{a} \mathbf{b} \) and can be expressed as the sum of a scalar and a wedge product:

\[
\mathbf{a} \mathbf{b} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b}
\]
Geometric Algebra  $G_{p,q,r}$

In general a geometric algebra $G_{p,q,r}$ is a linear space of dimension $2^n$, where $n=p+q+r$.

Geometric Product

Let $e_i$ and $e_j$ be two orthonormal basis vectors of the vector space. Then the geometric product for these vectors is defined as:

$$e_i e_j = \begin{cases} 
1 & \text{for } i = j \in \{1,\ldots,p\} \\
-1 & \text{for } i = j \in \{p+1,\ldots,p+q\} \\
0 & \text{for } i = j \in \{p+q+1,\ldots,n\} \\
e_{ij} = e_i \wedge e_j = -e_j \wedge e_i & \text{for } i \neq j
\end{cases}$$

bivector
Example the geometric algebra $G_{4,1,0}$ has $2^5 = 32$ elements.

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>10</th>
<th>10</th>
<th>5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalar</td>
<td>Vectors</td>
<td>Bivectors</td>
<td>Trivectors</td>
<td>Tetravectors</td>
<td>Pentavectors or Pseudoscalar</td>
</tr>
</tbody>
</table>

\[ e_1^2 = e_2^2 = e_3^2 = e_4^2 = 1 \]
\[ e^2 = -1 \]
\[ e_0 = \frac{(e_- - e_+)}{2}, \]
\[ e_\infty = e_- + e_+, \]
\[ I_c = e_1 \wedge e_2 \wedge e_3 \]
\[ E = e_4 \wedge e_5 \]
\[ I_c = I_e \wedge E \]

Note:
\[ e_4 = e_+ \]
\[ e_5 = e_- \]

\[ x = x_e + \frac{1}{2} x_e^2 e_\infty + e_o \]
\[ x^2 = 0 \]
\[ A \cdot B = -\frac{1}{2} (a - b)^2 \]

Conformal Geometric Algebra for 3D space

3D $\rightarrow$ 5D

Metric recovery !!
Conformal Entities

\[ s = p_c - \frac{1}{2} \rho^2 e_\infty \]
\[ x_c \cdot s = 0 \]

\[ S^* = S \cdot I_c \]

\[ S^* = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \]
\[ x_c \wedge S^* = 0 \]

\[ \pi^* = \pi \cdot I_c \]

\[ \pi = n - de_\infty \]

\[ \pi^* = x_1 \wedge x_2 \wedge x_3 \wedge e_\infty \]
Conformal Entities

\[ Z^\ast = Z \cdot I_c \]

Sphere, circle, pair of points and point

\[ z = s_1 \land s_2 \]

\[ l^\ast = l \cdot I_c \]

\[ l = \pi_1 \land \pi_2 \]
Inversion and Dilation

The reflection of the point in an sphere.

\[ s = e_o - \frac{1}{2} e_\infty \]
\[ \rho x' = -sx s^{-1} \]

The dilation is the product of two reflections

\[ x'' = (e_o - \frac{1}{2} \rho^2 e_\infty)(e_o - \frac{1}{2} e_\infty)x(e_o - \frac{1}{2} e_\infty)(e_o - \frac{1}{2} \rho^2 e_\infty) \]
\[ D_\rho = (e_o - \frac{1}{2} \rho^2 e_\infty)(e_o - \frac{1}{2} e_\infty) \]
\[ D_\rho = e^{-\frac{1}{2}Eln(\rho)} \]
\[ x' = D_\rho x D_\rho^{-1} \]

\[ s_1 = e_o - \frac{1}{2} e_\infty \]
\[ s_2 = e_o - \frac{1}{2} \rho^2 e_\infty \]
Rotor, Translator and Motor

Translator

\[ T = \left( 1 + \frac{t}{2} I \right) \]

Rotation in an arbitrary axis:

\[ R_s = TR_iT^{-1} \]

\[ R_s = e^{\left((1+\frac{t}{2}e_{\alpha})(-\frac{q}{2}I)(1+\frac{t}{2}e_{\alpha})\right)} = e^{\left(-\frac{q}{2}(I+e(t\cdot I))\right)} \]

\[ R_s = e^{-\frac{q}{2}L} \]

The Motor (moment-vector):

\[ M = T_s R_s \]

\[ M = \left(1 + \frac{t_s}{2} I\right)R_s \]

\[ O' = M O \tilde{M} \]

A motor for rigid motion of geometric objects (like points, lines, planes, sphere)
Mobius transformation result as the projection of simple combination of motions on the sphere.
Conformal transformation in geometric algebra

\[ x \xrightarrow{\text{inversion}} \frac{x}{x^2} \xrightarrow{\text{translation}} \frac{x}{x^2} + a \]

\[ \xrightarrow{\text{inversion}} \frac{x}{x^2} + a \]

\[ \frac{x}{x^2} + a = \frac{x + ax^2}{1 + 2a \cdot x + a^2 x^2} = x \frac{1}{1 + ax} \]

In conformal geometry

\[ x' = K_a x \vec{K}_a = e \{ T_a (exe) \vec{T}_a \} e \]

\[ x \mapsto x' = x \frac{1}{1 + ax} \Rightarrow x_c = F(x') = F(K_a x \vec{K}_a) = F(e \{ T_a (exe) \vec{T}_a \} e) = x' + \frac{1}{2} x'^2 e_\infty + e_o \]

\[ x \in \mathbb{R}^3 \]

\[ x_c = F(x) = x_e + \frac{1}{2} x_e^2 e_\infty + e_o \in \mathbb{R}^5 \]
Forward kinematics

\[ x = Rx + t \quad \Rightarrow \quad X' = MX\tilde{M} \]

Rotation in an arbitrary axis

\[ M = TR_lT^{-1} \]

\[ M = e^{((1+\frac{t}2)e_\infty)(-\frac{\theta}2l)(1+\frac{t}2e_\infty))} \]

\[ M = e^{-\frac{\theta}2L} \]

\[ \tilde{M} = e^{\frac{\theta}2L} \]

Motor (dual quaternion) better operator for rigid transformation representation than a matrix representation \((R, t)\)

\[ x'_p = \prod_{i=1}^{n} M_i x_p \prod_{i=1}^{n} \tilde{M}_{n-i+1} \]
Differential kinematics

\[ Q' = \prod_{i=1}^{n} M_i Q \prod_{i=1}^{n} \tilde{M}_{n-i+1} \]

\[ x'_{p} = \prod_{i=1}^{n} M_{i} x_{p} \prod_{i=1}^{n} \tilde{M}_{n-i+1} \]

\[ dx'_{p} = \sum_{j=1}^{n} \partial_{q_{j}} \left( \prod_{i=1}^{n} M_{i} x_{p} \prod_{i=1}^{n} \tilde{M}_{n-i+1} \right) dq_{j} \]

which can be further simplified as

\[ dx'_{p} = -\frac{1}{2} \sum_{j=1}^{n} \left[ \prod_{i=1}^{j-1} M_{i} \left( L_{j} \left( \prod_{i=j}^{n} M_{i} x_{p} \prod_{i=1}^{n-j+1} \tilde{M}_{n-i+1} \right) - \left( \prod_{i=j}^{n} M_{i} x_{p} \prod_{i=1}^{n-j+1} \tilde{M}_{n-i+1} \right) L_{j} \right) \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right] dq_{j} \]

Since \( M = e^{-\frac{1}{2} qL} \), the differential of the motor is \( d(M) = -\frac{1}{2} MLdq \), thus we can write the partial differential of the motor’s product as follows.

\[ \partial_{q_{j}} \left( \prod_{i=1}^{j} M_{i} \right) = -\frac{1}{2} \prod_{i=1}^{j} M_{i} L_{j} = -\frac{1}{2} \left( \prod_{i=1}^{j-1} M_{i} \right) L_{j} M_{j} \]

Similarly the differential of the \( \tilde{M} = e^{\frac{1}{2} qL} \) give us \( d(\tilde{M}) = \frac{1}{2} MLdq \) and the differential of the product is.

\[ \partial_{q_{j}} \left( \prod_{i=n-j+1}^{n} \tilde{M}_{n-i+1} \right) = \frac{1}{2} \tilde{M}_{j} L_{j} \prod_{i=n-j+2}^{n} \tilde{M}_{n-i+1} \]
Replacing both in previous equation

\[ dx'_p = \sum_{j=1}^{n} \left[ -\frac{1}{2} \prod_{i=1}^{j-1} M_i L_j M_{j-i} \prod_{i=j+1}^{n} M_i x_p \prod_{i=1}^{n} \tilde{M}_{n-i+1} + \frac{1}{2} \prod_{i=1}^{n} M_i x_o \prod_{i=1}^{n-j} \tilde{M}_{n-i+1} \tilde{M}_j L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right] dq_j, \]

which can be further simplified as

\[ dx'_p = -\frac{1}{2} \sum_{j=1}^{n} \left[ \prod_{i=1}^{j-1} M_i \left( L_j \left( \prod_{i=j}^{n} M_i x_o \prod_{i=1}^{n-j+1} \tilde{M}_{n-i+1} \right) - \left( \prod_{i=j}^{n} M_i x_p \prod_{i=1}^{n-j} \tilde{M}_{n-i+1} \tilde{M}_j L_j \right) \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right) \right] dq_j \]

Note that the product of a vector with an \( r \)-vector is given by

\[ a \cdot B_r = \frac{1}{2} \left( a B_r + (-1)^{r+1} B_r a \right) \]

As \( L \) is a bivector and \( x_p \) is a vector

\[ dx'_p = \sum_{j=1}^{n} \left[ \prod_{i=1}^{j-1} M_i \left( \left( \prod_{i=j}^{n} M_i x_p \prod_{i=1}^{n-j+1} \tilde{M}_{n-i+1} \right) \cdot L_j \right) \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right] dq_j \]

Similar as the case of points all the transformations in conformal geometric algebra can be also applied to the lines,

\[ dx'_p = \sum_{j=1}^{n} \left[ \left( \prod_{i=1}^{j-1} M_i \prod_{i=j}^{n} M_i x_p \prod_{i=1}^{n-j+1} \tilde{M}_{n-i+1} \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right) \cdot \left( \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right) \right] dq_j \]
Differential kinematics

Similar as the case of points all the transformations in conformal geometric algebra can be also applied to the lines, thus

\[ dx'_p = \sum_{j=1}^{n} \left[ \left( \prod_{i=1}^{j-1} M_i \prod_{i=j}^{n} M_i x_p \prod_{i=1}^{n-j+1} \tilde{M}_{n-i+1} \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right) \cdot \left( \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right) \right] dq_j \]

Since \( \prod_{i=1}^{j-1} M_i \prod_{i=j}^{n} M_i = \prod_{i=1}^{n} M_i \) we have

\[ dx'_p = \sum_{j=1}^{n} \left[ \left( \prod_{i=1}^{n} M_i x_p \prod_{i=1}^{n} \tilde{M}_{n-i+1} \right) \cdot \left( \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right) \right] dq_j \]

Since \( x'_p = \prod_{i=1}^{n} M_i x_p \prod_{i=1}^{n} \tilde{M}_{n-i+1} \)

\[ dx'_p = \sum_{j=1}^{n} \left[ x'_p \cdot \left( \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right) \right] dq_j \]

If we define \( L' \) as function of \( L \) as follows

\[ L'_j = \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i}, \]

we get a very compact expression of differential kinematics.

\[ dx'_p = \sum_{j=1}^{n} \left[ x'_p \cdot L'_j \right] dq_j, \]

in this way we can finally write:

\[ \dot{x}'_p = \begin{pmatrix} x'_p \cdot L'_1 & \cdots & x'_p \cdot L'_n \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \vdots \\ \dot{a}_n \end{pmatrix} \]
Robot Dynamics

\[ K = \frac{1}{2} \dot{q}^T \left(V^T m V + \delta I \right) \dot{q} \]

\[ U = \sum_{i=1}^{n} x_i' \cdot F_i. \]

Lagrange's Equations

\[ \frac{\partial L}{\partial \dot{q}} = \frac{\partial}{\partial \dot{q}} \left( \frac{1}{2} \dot{q}^T \left(V^T m V + \delta I \right) \dot{q} \right) = (V^T m V + \delta I) \ddot{q}. \]

\[ C = V^T m \dot{V}. \]

\[ (V^T m V + \delta I) \ddot{q} + V^T m \dot{V} \dot{q} + V^T F = \tau, \]

\[ \delta I \ddot{q} + V^T (m V \ddot{q} + m \dot{V} \dot{q} + F) = \tau. \]

\[ M \ddot{q} + C \ddot{q} + G = \tau \]

\[ \delta I \ddot{q} + V^T m (V \ddot{q} + \dot{V} \dot{q} + a) = \tau. \]

geometric algebra \( G_{4,1,0} \).
Grad, div, curl and blade-Laplacian

\[ \nabla F = \nabla \cdot F + \nabla \wedge F \]

\[ \nabla \cdot F = \frac{\partial}{\partial^k} e^k \cdot F = \frac{\partial F^k}{\partial^k} = \partial_k F^k \]

\[ \nabla \wedge F = e \nabla (\partial_i F) = e^i \wedge e^j \partial_i F_j. \quad \rightarrow \quad \nabla \wedge F = I_3 \nabla \times F. \]

Laplacian

\[ \partial^2 = \partial \cdot \partial = \nabla^2, \]

Given \( F \) a multivector valued function, the blade-Laplace operator of \( F \) is given by

\[ \Delta_L F = \partial \cdot (\partial \wedge F) + \partial \wedge (\partial \cdot F) \]

\[ = (\partial \cdot \partial) F + (\partial \wedge \partial) \times F = \partial^2 F + (\partial \wedge \partial) \times F. \]

\[ \Delta_L F = \Delta_M + (\partial \wedge \partial) \times F. \]

\[ \Delta_L F = \Delta_M + S(\dot{\partial}) \times \dot{F}. \]
Long term research goal

Robotics
Neuroscience
materials science

Brain coding
neocortex

sensor stimuli

Hyperspherical retina
Ego coordinates
fusion, storing
learning
control
action

World coordinates
Geometric Cybernetics
Development of algorithms for PAC systems

- Body-sensors calibration
- RGB-d cameras
  3D reconstruction
  navigation
- Learning
  Geometric control
- Perception
  Image Processing
  Computer Vision
- Model human vision
- Humanoid
- Grasping, object manipulation
Body-sensor calibration: relate the coordinate systems of the sensors with the robot body
Hand-eye calibration in CGA

\[ AX = XB \Rightarrow \begin{bmatrix} R_A & t_A \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_X & t_X \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_X & t_X \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_B & t_B \\ 0 & 1 \end{bmatrix} \]

Using motors

\[ M_A M_X = M_X M_B \]

However, according Chen’s theorem the only what matters is the relative motion of the involved screw lines
Hand-eye calibration

The angle and pitch of the gripper (or endoscope) are equal to the angle and pitch of the camera, they remain invariant under coordinate transformations, which is known as the Chen’s screw congruence theorem; therefore, the problem is solved using only the screw axes lines defined by the motors.

$$M_A M_X = M_X M_B$$

$$M = \cos\left(\frac{\theta}{2} + e_\infty \frac{d}{2}\right) + \sin\left(\frac{\theta}{2} + e_\infty \frac{d}{2}\right)l$$

$$L_A = a + a' = M_X L_B \tilde{M}_X$$

$$= (R + R')(b + b')(R + R')$$

$$= Rb\tilde{R} + e_\infty (Rb\tilde{R}' + Rb'\tilde{R} + R'b\tilde{R})$$

$$a = Rb\tilde{R}$$

$$a' = Rb\tilde{R}' + Rb'\tilde{R} + R'b\tilde{R}$$

Lines can be represented using bivectors $a$ and $a'$.

Multiplying from the right by $R$ and using the relationship $\tilde{R}R' + \tilde{R}'R = 0$, the following relationships are obtained:

$$aR - Rb = 0$$

$$(a' R - Rb') + (aR' - R'b) = 0$$

which can be expressed in matrix form as:

$$\begin{bmatrix}
  a - b & [a + b]_\times & 0_{3 \times 1} & 0_{3 \times 3} \\
  a' - b' & [a' + b']_\times & a - b & [a + b]_\times
\end{bmatrix}
\begin{bmatrix}
  R \\
  R'
\end{bmatrix} = 0$$
Hand-eye calibration

\[
\begin{bmatrix}
 a - b & [a + b]_x & 0_{3 \times 1} & 0_{3 \times 3} \\
 a' - b' & [a' + b']_x & a - b & [a + b]_x
\end{bmatrix}
\begin{bmatrix}
 R \\
 R'
\end{bmatrix} = 0
\]

\[
D
\]

\[
C = \begin{bmatrix}
 D_1^T & D_2^T & D_3^T & D_4^T & \ldots
\end{bmatrix}^T
\]

\[
\begin{bmatrix}
 R \\
 R'
\end{bmatrix} = \alpha \begin{bmatrix}
 u_1 \\
 v_1'
\end{bmatrix} + \beta \begin{bmatrix}
 u_2 \\
 v_2'
\end{bmatrix}
\]

Taking into account the constraints

\[
R \tilde{R} = 1 \quad \text{and} \quad \tilde{R}R' + \tilde{R}'R = 0
\]

\[
\alpha^2 u_1^T u_1 + 2\alpha \beta u_1^T u_2 + \beta^2 u_2^T u_2 = 0
\]

\[
\alpha^2 u_1^T v_1 + \alpha \beta (u_1^T v_2 + u_2^T v_1) + \beta^2 u_2^T v_2 = 0
\]

\[
\mu = \alpha / \beta
\]

\[
\beta^2 (\mu^2 u_1^T u_1 + \mu (2u_1^T u_2) + u_2^T u_2) = 1
\]

which takes two solutions for \( \beta \).
The procedure is repeated for each axis.
Calibrated robot: 3D reconstruction and controlled navigation
\[ \vec{H}(s, t) = h_x(s, t)e_1 + h_y(s, t)e_2 + h_z(s, t)e_3 \]

\[ N(s, t) = \left( \frac{\partial \vec{H}(s, t)}{\partial s} \wedge \frac{\partial \vec{H}(s, t)}{\partial t} \right) I_e \]
Robot Manipulador

AdeptSix600 (6 DOF)

Barrett Hand (4 DOF)
Grasping: three contact points

Three contact points: the forces meet at the center of mass.

Three contact points: Forces are parallel.

Multiple contact points.
Visually guide object grasping

- Segmentation
- Tracking
- Pose estimation
- Grasping
Humanoid Mechanism and Perception
Long term research goal

- Robotics
- Neuroscience
- Materials science

Mathematical framework

Brain coding
- Neocortex

Sensor stimuli

Cognitive Architecture
- Ego coordinates
- World coordinates
- Fusion, storing
- Learning
- Control
- Action

Motivation
Humanoid walking

\[ L_1 = e_{23} + e_{\infty} (o_{1} \cdot e_{23}), \]
\[ L_2 = e_{12} + e_{\infty} (o_{2} \cdot e_{12}), \]
\[ L_3 = e_{31} + e_{\infty} (o_{3} \cdot e_{31}), \]
\[ L_4 = e_{23} + e_{\infty} (o_{4} \cdot e_{23}), \]
\[ L_5 = e_{23} + e_{\infty} (o_{5} \cdot e_{23}), \]
\[ L_6 = e_{12} + e_{\infty} (o_{6} \cdot e_{12}), \]
\[ L_7 = e_{23} + e_{\infty} (o_{7} \cdot e_{23}), \]
\[ L_8 = e_{12} + e_{\infty} (o_{8} \cdot e_{12}), \]
\[ L_9 = e_{31} + e_{\infty} (o_{9} \cdot e_{31}), \]
\[ L_{10} = e_{23} + e_{\infty} (o_{10} \cdot e_{23}), \]
\[ L_{11} = e_{23} + e_{\infty} (o_{11} \cdot e_{23}), \]
\[ L_{12} = e_{12} + e_{\infty} (o_{12} \cdot e_{12}). \]

**Figure 13.** Control signals (joint torques) for the six joints of the left leg (*super twisting* technique).
Human versus robot vision system

Motivation

Use a stochastic geometric framework for modeling biologic concepts and develop algorithms for intelligent mechatronic systems.

perception

Signal flow

Visual cortex
Lie Algebra of visual perception

Hoffman L. Model of Human Vision system using Lie algebras 1960's

More complex combinations
the first ~14 years of life
Combinations of Lie operators for visual flow

\[
\mathcal{L}_i = \frac{\partial}{\partial t} \mathcal{L}_{M1} = x \frac{\partial}{\partial x}, \quad \mathcal{L}_{M2} = y \frac{\partial}{\partial y}, \quad \mathcal{L}_{M3} = \frac{\partial}{\partial x} - x \frac{\partial}{\partial y} = \mathcal{L}_M,
\]

\[
\mathcal{L}_{M1} = l \frac{\partial}{\partial x} + x \frac{\partial}{\partial t}, \quad \mathcal{L}_{M2} = l \frac{\partial}{\partial y} + y \frac{\partial}{\partial t}, \quad \mathcal{L}_{M3} = x \frac{\partial}{\partial t} - y \frac{\partial}{\partial x} = \mathcal{L}_M.
\]

<table>
<thead>
<tr>
<th>Name / Operator</th>
<th>Funktion ( F(x,y) = )</th>
<th>Bild (für ( w = \sin ))</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Einheitliche Intensität</strong></td>
<td>( \mathcal{L}_e = \frac{\partial}{\partial x} + \frac{\partial}{\partial y} )</td>
<td>( c ) (Konstante)</td>
</tr>
<tr>
<td><strong>Horizontale Translation</strong></td>
<td>( \mathcal{L}_e = \frac{\partial}{\partial x} )</td>
<td>( w(x) )</td>
</tr>
<tr>
<td><strong>Vertikale Translation</strong></td>
<td>( \mathcal{L}_e = \frac{\partial}{\partial y} )</td>
<td>( w(y) )</td>
</tr>
<tr>
<td><strong>Schiefwinklige Translation</strong></td>
<td>( \mathcal{L}_e = \frac{\partial}{\partial x} + m \frac{\partial}{\partial y} )</td>
<td>( w(y - mx) )</td>
</tr>
<tr>
<td><strong>Rotation</strong></td>
<td>( \mathcal{L}_e = -(y - yc) \frac{\partial}{\partial x} + (x - xc) \frac{\partial}{\partial y} )</td>
<td>( w(\sqrt{(x-xc)^2 + (y-yc)^2}) )</td>
</tr>
<tr>
<td><strong>Dilatation/Kontraktion</strong></td>
<td>( \mathcal{L}_e = (x - xc) \frac{\partial}{\partial x} + (y - yc) \frac{\partial}{\partial y} )</td>
<td>( w(\text{arcosh}(\frac{y}{x} - 1)) )</td>
</tr>
<tr>
<td><strong>Hyperbolische Rotation Typ A</strong></td>
<td>( \mathcal{L}_e = (x - xcb) \frac{\partial}{\partial x} + (y - ycb) \frac{\partial}{\partial y} )</td>
<td>( w(\sqrt{(x-xcb)^2 + (y-ycb)^2}) )</td>
</tr>
<tr>
<td><strong>Hyperbolische Rotation Typ b</strong></td>
<td>( \mathcal{L}_e = (y - ycb) \frac{\partial}{\partial x} + (x - xcb) \frac{\partial}{\partial y} )</td>
<td>( w(\sqrt{(x-xcb)^2 + (y-ycb)^2}) )</td>
</tr>
</tbody>
</table>
Binocular vision

Human visual cortex has some intrinsic manner (subgroup of the affine group) to convert this binocular view of the world into a cyclopean view (without losing the 3D scene).

- The Superior Colliculus contains a motor map; activity in a given region correlates with a particular a saccade-vector.
- The saccade-vectors are independent of the absolute gaze direction.

E.L. Schwartz, V1-V2-V3

Quasi-conformal. Mapping
(log of Mobiüs transformtion)
Fixation, Stereopsis and Vergence

Moving an object slightly off the horopter creates a small retinal disparity that stimulates stereoscopic depth perception.

Source: Enright, 1983.

Source: Bell, 1823.
Listings law using rotors and translators

Pure rotations

Using rotors

\[ S(b, a) = S(b)S(a)^{-1} = (1 + bc_3)(1 - c_3a) \]

Parameterization of plane deformation

Motor to Include small displacements

\[ \phi'_d = M \phi_d \dot{M} - R_s n \dot{R}_s - l(d + (R_s n \dot{R}_s) \cdot t_s) \]

Listings planes of a binocular system

Tracking of moving object
The computation of the motion axes is done with motors $M$ in $G_{4,1,0}$.

\[
L_A = M L B \tilde{M} \\
L_A = n_A + e_\infty m_A \\
L_B = n_B + e_\infty m_B \\
R = e^{\frac{1}{2} \theta B} = \cos(\theta B) + B \sin(\frac{\theta B}{2}) \\
T = e^{\frac{1}{2} t \tilde{e}} = 1 + \frac{1}{2} t \tilde{e}
\]

\[
\begin{bmatrix}
  a - b & [a + b]_x & 0_{3 \times 1} & 0_{3 \times 3}, \\
  a' - b' & [a' + b']_x & a - b & [a + b]_x
\end{bmatrix}
\begin{bmatrix}
  R \\
  R'
\end{bmatrix}
\]
Conformal Model of Humanoid robot Vision

Stereo Rig, Azimuth Plane, 3D Reconstruction

Duals of poncelet points

Spherical representation of the 3D visual space
Algebra of spheres for self-localization

Applications

Model-based Humamoid self-localization

Recognition

Location hypotheses generation mechanism

Inverse kinematics

Model-based Humamoid self-localization
Simplex in conformal GA for 3D reconstruction

Delanauy triangulation

 simplex

Applications

\( a \land b \land c \land e \)
This algorithm computes the axes of rotation based on measurements of lines taken from the real world.
The procedure is repeated for each axis.
Defining the orientation error

\[ e_A = A - A_d \]

We obtain the orientation error dynamics as follows

\[ \dot{e}_A = J_A \dot{\theta} - \dot{A}_d ; \text{where } J_A = \frac{\partial A(\theta)}{\partial \theta} \]

So we need to use a robust control technique, because of the unknown term \( \dot{A}_d \).

Integral Sliding Modes (ISM) is a control strategy having the following advantages.

✓ Robustness against parameter uncertainty and external disturbances.
✓ Robustness starting from the initial time instant.
✓ Signal control low gain (Magnitude).
✓ Chattering-free, pre-filtering the discontinuous control term.
Designing ISM Controller

Assuming joint velocities as control signals, we can re-define the error dynamics as:

\[ \dot{e}_A = J_A U - \dot{A}_d; \quad \text{where } U = U_1 + U_2 \]

Using U1 to control “ideal” dynamics, without unknown and disturbance terms, and U2 to reject these terms. We define the sliding surface as:

\[ S = e_A + z \]

where z is an auxiliar variable. Then its dynamic is given by:

\[ \dot{S} = J_A (U_1 + U_2) - \dot{A}_d + z \]

Defining \( z = -J_A U_1 \) we obtain

\[ \dot{S} = J_A U_2 - \dot{A}_d \]

then with \( U_1 = -K_1 J_A^+ e_A \) a stable error system is obtained \((e_A \to 0)\)

and with \( U_2 = -K_2 J_A^+ \text{sign}(S) \) we reject the disturbance terms,

where \( K_1 > 0 ; \ K_2 > \| \dot{A}_d \| \).
Comparison of controllers

**Figure A) PID**

**Figure B) ISM**
Estimation of Pose and 3D Structure

Estimation of the orientation and pose $M = RT$ of the cameras using Kalman filter techniques

For

Oculomotor control for: vergence and tracking

Estimation von $M_l = R_lT_l$ and $M_r = R_rT_r$

Bonus:
--independency of errors in sensor-Body calibration.
--3D reconstruction and relocalization
1. Egomotion and 3D reconstruction: object searching

Hand-eye calibration is inaccurate due uncertainties. Free coordinates egomotion estimation helps for a better self-localization and 3D reconstruction.

Using Kalman filter techniques the rotor $R$ and the translator $T$ are estimated. Cameras work in a master slave fashion by fixation.

Cyclopean eye 
Vieht-Müller circles and fixation points

Camera was moved by hand in a 3D trajectory

Estimated rotation angles 
Of left and right cameras

Locus of ZOD
1. Egomotion and 3D reconstruction: object recognition and pose estimation

Fixation point to get stereopsis for better correspondences and feature extraction.

Focus of Attention

Applications

Object detection

Features computing (integral images)

AdaBoost (Selects a small number of critical visual features)

Cascade method (Combining more complex classifiers)

Object recognition and pose estimation

Grasping !!
Egomotion and 3D reconstruction

view staircase for climbing
Egomotion and 3D reconstruction

Views by changing the vergence
Quaternion Spike Neurons
Quatérmion Spike Neural Network

$$\Delta W_{ml}^n = -\eta \frac{\partial E}{\partial W_{ml}^n}$$

extendiendo la ecuación anterior

$$\frac{\partial E}{\partial W_{ml}^n} = \frac{\partial E}{\partial W_{ml}^n} + \frac{\partial E}{\partial W_{ml}^n} i + \frac{\partial E}{\partial W_{ml}^n} j + \frac{\partial E}{\partial W_{ml}^n} k$$

utilizando la regla de la cadena

$$\begin{align*}
\frac{\partial E}{\partial W_{ml}^n} &= \frac{\partial E}{\partial \tau_{ml}^n} \frac{\partial \tau_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial \sigma_{ml}^n} \frac{\partial \sigma_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial F_{ml}^n} \frac{\partial F_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial k_{ml}^n} \frac{\partial k_{ml}^n}{\partial W_{ml}^n} \\
\frac{\partial E}{\partial W_{ml}^n} &= \frac{\partial E}{\partial \tau_{ml}^n} \frac{\partial \tau_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial \sigma_{ml}^n} \frac{\partial \sigma_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial F_{ml}^n} \frac{\partial F_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial k_{ml}^n} \frac{\partial k_{ml}^n}{\partial W_{ml}^n} \\
\frac{\partial E}{\partial W_{ml}^n} &= \frac{\partial E}{\partial \tau_{ml}^n} \frac{\partial \tau_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial \sigma_{ml}^n} \frac{\partial \sigma_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial F_{ml}^n} \frac{\partial F_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial k_{ml}^n} \frac{\partial k_{ml}^n}{\partial W_{ml}^n} \\
\frac{\partial E}{\partial W_{ml}^n} &= \frac{\partial E}{\partial \tau_{ml}^n} \frac{\partial \tau_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial \sigma_{ml}^n} \frac{\partial \sigma_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial F_{ml}^n} \frac{\partial F_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial k_{ml}^n} \frac{\partial k_{ml}^n}{\partial W_{ml}^n} \\
\frac{\partial E}{\partial W_{ml}^n} &= \frac{\partial E}{\partial \tau_{ml}^n} \frac{\partial \tau_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial \sigma_{ml}^n} \frac{\partial \sigma_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial F_{ml}^n} \frac{\partial F_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial k_{ml}^n} \frac{\partial k_{ml}^n}{\partial W_{ml}^n} \\
\frac{\partial E}{\partial W_{ml}^n} &= \frac{\partial E}{\partial \tau_{ml}^n} \frac{\partial \tau_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial \sigma_{ml}^n} \frac{\partial \sigma_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial F_{ml}^n} \frac{\partial F_{ml}^n}{\partial W_{ml}^n} + \frac{\partial E}{\partial k_{ml}^n} \frac{\partial k_{ml}^n}{\partial W_{ml}^n}
\end{align*}$$

$$\frac{\partial E}{\partial W_{ml}^n} = \frac{E(r + ti + tj + tk) + E(-ti + tr + tj + tk)}{\sum_l W_l \frac{\partial E}{\partial W_{ml}^n}} + \frac{E(-tj + tk + ti + ty + tk)}{\sum_l W_l \frac{\partial E}{\partial W_{ml}^n}} + \frac{E(-tk + tj + ti + tr + tk)}{\sum_l W_l \frac{\partial E}{\partial W_{ml}^n}}$$

$$\frac{\partial E}{\partial W_{ml}^n} = \left( E \otimes \frac{1}{\sum_l W_l \frac{\partial E}{\partial W_{ml}^n}} \right) \otimes s^*$$

$$W_{ml}^n(K + 1) = W_{ml}^n(K) + \eta \frac{\partial E(K)}{\partial W_{ml}^n}$$
Quaternion Spike Neurons

Control of a 6 DOF robot manipulator
Neuronavigator

Imágenes TAC (tomografía axial computarizada)

Imágenes IRM (resonancia magnética) e IRMF (resonancia magnética funcional)

Atlas anatómico

Registro no lineal (alineamiento)

Ciclo iterativo

Clasificación estadística y segmentación

Análisis funcional (IRMF)

Modelo 3D preoperatorio

Registro (Alineamiento)

Modelo 3D intraoperatorio

Monitoreo en tiempo real del instrumento quirúrgico

Realidad virtual aumentada

Endoscopia real y virtual

Señalización ecográfica, de corrimientos del tumor
Segmentation

- Neural net to adjust a topological map to the object we are interested in → object contour

- Gradient Vector Flow → input selection (for the NN) and in the adaptation process.

- Geometric Algebra → each neural unit (neuron) has an associated transformation, which is expressed as a *motor* (rotation-translation) in the GA
  - All the motors of the net determine the object shape (when they are applied to a specific point)
Gradient Vector Flow (GVF)

- Objective: spread the gradient vectors
- GVF is a vector field
  \[ \mathbf{v}(x, y) = \begin{bmatrix} u(x, y) & v(x, y) \end{bmatrix} \]
- Minimizes
  \[ \varepsilon = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |\mathbf{v} - \nabla f|^2 \, dx \, dy \]
- where \( f(x, y) \) is an image borders map

| \nabla f | Big value
| \nabla f | \approx 0
Generalized Gradient Vector Flow (GGVF)

- Change the coefficient $\mu$ and $|\nabla f|^2$ by the weight functions

$$g(|\nabla f|) = e^{-\frac{|\nabla f|}{\mu}}$$

$$h(|\nabla f|) = 1 - g(|\nabla f|)$$

$$\mathcal{E} = \iint g(|\nabla f|) \nabla^2 \mathbf{v} - h(|\nabla f|) |\mathbf{v} - \nabla f|^2 \, dx \, dy$$
Generalized Gradient Vector Flow (GGVF)
Input image \(\rightarrow\) Compute the vector field \(\rightarrow\) Compute streamlines \(\rightarrow\) Determine the inputs to the net

Training of the net \(\rightarrow\) GNG

Set of versors \(\rightarrow\) Apply versors to selected point to define the object shape

Conformal Geometric Algebra

Training of the net GNG

Set of versors

Apply versors to selected point to define the object shape

Segmented image or 3D object
Algorithm (2D) to adjust the topologic map

- Every certain number $\lambda$ of iterations, insert new neurons
  - Find the neuron $n_i$ with the highest value of $rsf$. If any of its neighbors, say $n_j$, is connected with an edge larger than $c_{max}$, then create a new neuron $n_{new}$ between $n_i$ and $n_j$ with
    
    $$T_{new} = \frac{T_i + T_j}{2}, \quad v_{l_{new}} = \frac{v_i + v_j}{2}$$

  - Erase old connections between $n_i$ and $n_j$, and create new connections (edges) from $n_{new}$ to $n_i$ and $n_j$

- Repeat previous steps

- By this way, we are minimizing the quantification error
  
  $$\chi = \sum_{\forall \varsigma} (((M_{\varsigma} P_0 \tilde{M}_{\varsigma}) - X_{\varsigma})^2)$$
Input image

Compute GGVF

Compute streamlines

Determine inputs to the net

Neural net

Apply $T_i$ and to extract the object shape

Output: Segmented image
Without GGVF

Quantification error: 1.19
Topographic error: 0.142

Quantification error: 1.10
Topographic error: 0.153

Quantification error: 2.04
Topographic error: 0.133

With GGVF

Quantification error: 0.98
Topographic error: 0.05

Quantification error: 0.23
Topographic error: 0.0

Quantification error: 1.002
Topographic error: 0.0
Inputs to the net and initialization
Segmentation

Input image  
GGVF  
Streamlines

Inputs to the net and initialization

Result
Advantages against GGVF-Snake
Segmentation
Algorithm 3D to adjust the topologic map

- Every certain number $\lambda$ of iterations, insert new neurons
  - Find the neuron $n_i$ with the highest value of $rsf$. If any of its neighbors, say $n_j$, is connected with an edge larger than $c_{\text{max}}$, then create a new neuron $n_{\text{new}}$ between $n_i$ and $n_j$ with
    - Erase old connections between $n_i$ and $n_j$, and create new connections (edges) from $n_{\text{new}}$ to $n_i$ and $n_j$

- Repeat previous steps

- By this way, we are minimizing the quantification error

$$
\chi = \sum_{\forall \varsigma}(((M_\varsigma P_0 \tilde{M}_\varsigma) - X_\varsigma)^2)
$$
Surface modeling using Neural Nets

Result with 100 neurons

Result with 170 neurons
Results with volumetric data

Inputs

Result with 100 neurons

Result with 200 neurons

Result with 300 neurons
3D Case Models using circles / spheres
Marching Spheres
Models based on spheres

\[ S^j_{p_i} = c_{p_i} + \frac{1}{2}(c_{p_i} - \rho_{p_i}^2)e_\infty + e_0 \]

\[ S^j_{m_i} = c_{m_i} + \frac{1}{2}(c_{m_i} - \rho_{m_i}^2)e_\infty + e_0 \]

\[ S^j_{g_i} = c_{g_i} + \frac{1}{2}(c_{g_i} - \rho_{g_i}^2)e_\infty + e_0 \]
**Tumor Model based on spheres**

<table>
<thead>
<tr>
<th>n / d</th>
<th>Number of spheres with each approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UoS – DT</td>
</tr>
<tr>
<td>3370 / 1</td>
<td>13480</td>
</tr>
<tr>
<td>3370 / 3</td>
<td>8642</td>
</tr>
</tbody>
</table>
Motor Interpolation of points, lines, planes, spheres

\[ \text{MSLERP}(t; M_i, M_f) = M_i(M_i^{-1}M_f)^t, \]
\[ (M_i^{-1}M_f)^t = \cos(t(\frac{\theta}{2} + e_\infty \frac{t}{2})) + L\sin(t(\frac{\theta}{2} + e_\infty \frac{t}{2})) \]
\[ = \cos(t(\frac{\hat{\theta}}{2})) + L\sin(t(\frac{\hat{\theta}}{2})), \]
\[ x_c^i = M_B x_c^{i-1} \tilde{M}_B^i \]
\[ L^i = M_B L^{i-1} \tilde{M}_B^i \]
\[ \pi^i = M_B \pi^{i-1} \tilde{M}_B^i \]
\[ s^i = D^i M_B s^{i-1} \tilde{M}_B^i D^{-1i} \]
Motor interpolation of the pose
Real time interpolation
Quaternionic Phase

\[ q = a + bi + cj + dk, \quad |q| = 1 \]

\[ \psi = -\frac{\arcsin(2(bc-ad))}{2} \]

<table>
<thead>
<tr>
<th>( \psi \in ]-\frac{\pi}{4}, \frac{\pi}{4}[ )</th>
<th>( \psi = \pm\frac{\pi}{4} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi = \frac{\text{arg}_i(q\beta(q))}{2} )</td>
<td>if ( \psi = \pm\frac{\pi}{4} ) choose ( \phi = 0 )</td>
</tr>
<tr>
<td>( \theta = \frac{\text{arg}_i(\alpha(q)q)}{2} )</td>
<td>( \theta = \frac{\text{arg}_i(\gamma(q)q)}{2} ) or ( \phi = \frac{\text{arg}_i(q\gamma(\bar{q}))}{2} )</td>
</tr>
</tbody>
</table>

if \( e^{i\phi}e^{k\psi}e^{i\theta} = -q \)
and \( \phi \geq 0 \)

\( \phi \to \phi - \pi \)

if \( e^{i\phi}e^{k\psi}e^{i\theta} = -q \)
and \( \phi < 0 \)

\( \phi \to \phi + \pi \)
Quaternion Wavelet Transform

QWT Is a natural extension of the real and complex wavelet transforms.
Quaternion Gabor Filters

Quaternion Gabor Filter

\[ G(x) = e^{-x^2/2\sigma_1^2 - y^2/2\sigma_2^2} e^{i(c_1 x/\sigma_1)} e^{j(c_2 y/\sigma_2)} \]

Low-Pass (G)

High-Pass (H)
Quaternion Gabor Wavelets with selective orientations

15  45  75  -75  -45  -15  degrees
Quaternionic Wavelet Pyramid

Level 1

Rows

Columns

h₁

g₁

h₁

g₁

↓2 Aproximation

↓2 Horizontal

↓2 Vertical

↓2 Diagonal

↓2 subsampling

Level 2

Rows

Columns

h₂

g₂

h₂

g₂

↓2 Aproximation

↓2 Horizontal

↓2 Vertical

↓2 Diagonal

↓2 subsampling
Motion Detection

Horizontal and Vertical disparity is calculated with the formulas

\[ d_x(x) = \frac{\phi_2(x) - \phi_1(x) + n(2\pi + k)}{2\pi u_{\text{ref}}}, \quad d_y(x) = \frac{\theta_2(x) - \theta_1(x) + m\pi}{2\pi v_{\text{ref}}} \]

With reference frequency

\[ u_{\text{ref}} = \frac{1}{2\pi} \frac{\partial \phi_1}{\partial x}(x), \quad v_{\text{ref}} = \frac{1}{2\pi} \frac{\partial \theta_1}{\partial y}(x) \]
A Confidence Matrix is calculated for each level as

\[
\text{Conf}_{h}(x) = C_{h}(k_{1}^{q}(x))C_{h}(k_{2}^{q}(x))
\]

\[
\text{Conf}_{v}(x) = C_{v}(k_{1}^{q}(x))C_{v}(k_{2}^{q}(x))
\]

With

\[
C_{h}(k^{q}(x)) = \begin{cases} 
1 & \text{if } \text{mod}_{i}(k^{q}(x)\beta(k^{q}(x))) > \tau \\
0 & \text{else.}
\end{cases}
\]

\[
C_{v}(k^{q}(x)) = \begin{cases} 
1 & \text{if } \text{mod}_{j}(\alpha(k^{q}(x))k^{q}(x)) > \tau \\
0 & \text{else.}
\end{cases}
\]
Motion Estimation Algorithm

Original Sequence

All the levels of the pyramid are combined

QWT

Disparity

Magnitude Approximation Phase Approximation Phase Approximation Phase y Approximation Magnitude Approximation Phase Approximation Phase Approximation Phase y Approximation Magnitude Approximation Phase Approximation Phase Approximation Phase y Approximation Magnitude Approximation Phase Approximation Phase Approximation Phase y Approximation Magnitude Approximation Phase Approximation Phase Approximation Phase y Approximation

Magnitude Vertical Phase x Vertical Phase y Vertical Phase y Vertical Phase x Vertical Phase y Vertical Phase x Vertical Phase y Vertical Phase x Vertical Phase y Vertical
Results
Results

Original Images

Confidence in 4 levels

Resulting Vectors
Combination

Disparity levels are combined

Horizontal Disparity

Vertical Disparity
An Experiment

Original Images

Resulting Vectors

Confidence in 4 levels
Comparison

Bernard Algorithm, level 4

Using the QWT, level 4
Atomic function

The atomic function \( up(x) \) is generated by infinite convolutions of rectangular impulses. The function \( up(x) \) has the following representation in terms of the Fourier transform.

\[
up(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \prod_{k=1}^{\infty} \frac{\sin(\nu 2^{-k})}{\nu 2^{-k}} e^{i\nu x} d\nu
\]

\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{up}(\nu) e^{i\nu x} d\nu.
\]

\[
up(x, y) = up(x)up(y)
\]
Quaternion analytic signal

\[ f_A^q(x, y) = f(x, y) + i f_{H_i}(x, y) + j f_{H_j}(x, y) + k f_{H_k}(x, y) \]

where \( f_{H_i}(x, y) = f(x, y) \star \frac{1}{\pi x} \) and \( f_{H_j}(x, y) = f(x, y) \star \frac{1}{\pi y} \) are the partial Hilbert transforms and \( f_{H_k}(x, y) = f(x, y) \star \frac{1}{\pi^2 xy} \) is the Hilbert transform,

\[ \psi = -\frac{\arcsin(2(bc - ad))}{2} \]

- If \( \psi \in ]-\frac{\pi}{4}, \frac{\pi}{4}[ \), then \( \phi = \frac{\text{arg}_i(q{T_j}(q))}{2} \) and \( \theta = \frac{\text{arg}_j(T_i(q)q)}{2} \).
- If \( \psi = \pm \frac{\pi}{4} \), then select either \( \phi = 0 \) and \( \theta = \frac{\text{arg}_j(T_k(q)q)}{2} \) or \( \theta = 0 \) and \( \phi = \frac{\text{arg}_i(q{T_k}(q))}{2} \).
- If \( e^{i\phi} e^{k\psi} e^{j\theta} = -q \) and \( \phi \geq 0 \), then \( \phi \to \phi - \pi \).
- If \( e^{i\phi} e^{k\psi} e^{j\theta} = -q \) and \( \phi < 0 \), then \( \phi \to \phi + \pi \).
Monogenic signal

Hilbert transform \((f_H(x) = \frac{1}{\pi x})\)

\[ f_A(f(x)) = f(x) + if_H(x). \]

\[ f_A(f(x)) = |A|e^{i\theta} \]

\[ f_M(x) = f(x) + (i, j)f_R(x) = f(x) + (i, j)f(x) \star \frac{x}{2\pi|x|^3}. \]

\[ \psi = \arctan \left( \frac{|(i, j)f_R \star f(x)|}{f(x)} \right) \]

\[ \theta = \arctan \left( \frac{jf_R \star f(x)}{if_R \star f(x)} \right) \]

\[ |f_M(x)| = \sqrt{(i, j)f_R^2 + f(x)^2} \]

Hilbert and Riesz Transforms using Atomic Function

Hilbert transform

\[ g_{\mathbf{H}_i} f(x, y) = f(x, y) \ast \left( \text{dup}(x, y)_x \ast -\frac{1}{\pi} \log(|x|) \right) \]

\[ g_{\mathbf{H}_j} f(x, y) = f(x, y) \ast \left( \text{dup}(x, y)_y \ast -\frac{1}{\pi} \log(|y|) \right) \]

\[ g_{\mathbf{H}_k} f(x, y) = f(x, y) \ast \left( \text{dup}(x, y)_{xy} \ast -\frac{1}{\pi^2} \log(|x|) \log(|y|) \right) \]

Riesz transform

\[ g_{\mathbf{R}}(x) = f(x) \ast \left( \text{dup}(x)_{xy} \ast -\frac{1}{2\pi(|x|)} \right) \]

Left a) \( \text{dup}(x, y)_{x,y} \); right b) \( \text{dup}(x, y)_y \)
Atomic Function Quaternion Hilbert transform

Image and 2D quaternionic phases $\phi$, $\theta$ and $\psi$
Atomic Function Riesz transform

<table>
<thead>
<tr>
<th>Image</th>
<th>Local phase</th>
<th>orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Local phase" /></td>
<td><img src="image3.png" alt="Orientation" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Local phase" /></td>
<td><img src="image6.png" alt="Orientation" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Local phase" /></td>
<td><img src="image9.png" alt="Orientation" /></td>
</tr>
</tbody>
</table>
Multi-scale Riesz transform for symmetry feature extraction

Atomic filters in spatial and frequency domains at two levels of the multiresolution pyramid.
Multi-scale Riesz transform for symmetry feature extraction

a) Image, edge image, symmetry image with centroid and main axis. b) Extracted feature signature: using (left) the symmetry image, (middle) the phase \{\psi \text{ and } (right) the phase } \theta\}. c) Feature signatures of binarised images.

Feature extraction
Conformal Geometric Algebra Voting Scheme.

- In computer vision, *perceptual organization* consists in obtaining meaningful representations from raw visual data.

> We have reformulated the Tensor Voting algorithm [2] using representations in terms of k-vectors of the Conformal Geometric Algebra, which allow us to extract any kind of geometric entities or flags from images.

- The essential components of this voting scheme can be summarized in two methodologies:
  - To represent information using CGA.
  - To communicate information via a voting process and a clustering technique.

Let $p_0$ be a pixel, then we want $F \cdot p_0 = 0$ the perceptual structure $F$, that satisfy:

In order to compute $F$, all tokens in the neighborhood of $p_0$ cast a vote. Each vote is codify as a salient geometric entity.

**Definition:** A salient geometric entity is a set of points together with a function, that assigns a scalar value to each element of the set.

$L = \{p_c : p_c \cdot l_c = 0\}$, $W : \mathbb{R}^m \rightarrow \mathbb{R}$

Using CGA $\mathbb{G}_{3,1}$ a salient line is represented by:

where $p_c$ and $l_c$ are a point, and a line, respectively, in conformal representation.
The geometric structure of
\[ l_i = -\frac{u_i}{|p_{ie}|} e_1 - \frac{v_i}{|p_{ie}|} + \frac{|p_{ie}|}{2} e_\infty : \]

where:

\[ W(u_i, v_i) = \exp \left( -\frac{s^2 + c\rho^2}{\sigma^2} \right) \]

\[ s = \frac{\theta d}{\sin \theta}, \quad \rho = \frac{2 \sin \theta}{d}, \quad d = \sqrt{-2p_{ic} \cdot p_{0c}}. \]
Voting process for extraction of circles and lines.

\[ C_{012} \wedge e_\infty = (l_1 \wedge l_2)^*, \quad W_{012} = W(u_1, v_1) + W(u_2, v_2) \]
\[ C_{013} \wedge e_\infty = (l_1 \wedge l_3)^*, \quad W_{013} = W(u_1, v_1) + W(u_3, v_3) \]
\[ C_{014} \wedge e_\infty = (l_1 \wedge l_4)^*, \quad W_{014} = W(u_1, v_1) + W(u_4, v_4) \]
\[ C_{023} \wedge e_\infty = (l_2 \wedge l_3)^*, \quad W_{023} = W(u_2, v_2) + W(u_3, v_3) \]
\[ C_{024} \wedge e_\infty = (l_2 \wedge l_4)^*, \quad W_{024} = W(u_2, v_2) + W(u_4, v_4) \]
\[ C_{034} \wedge e_\infty = (l_3 \wedge l_4)^*, \quad W_{034} = W(u_3, v_3) + W(u_4, v_4) \]
Voting process for extraction of circles and lines

- Use DBSCAN to group similar geometric entities.
- Compute the weighted mean of each cluster $i$:
  \[ C_x = \frac{\sum_i (W_i C_{xi})}{\sum_i W_i}, \quad C_y = \frac{\sum_i (W_i C_{yi})}{\sum_i W_i}, \quad \bar{\rho} = \frac{\sum_i (W_i \rho_i)}{\sum_i W_i}, \quad D_i = \sum_i W_i \]
- Select clusters with maximum density $D_i$.

Geometric Structure:
\[ l_i = -\frac{u_i}{|p_{ic}|} e_1 - \frac{v_i}{|p_{ic}|} e_2 + \frac{|p_{ic}|}{2} e_\infty \]

Perceptual Saliency:
\[ W(u_i, v_i) = \exp \left(-\frac{s^2 + \epsilon^2}{\sigma^2}\right) \]

*Geometric procedure:
\[ S_{ij} = l_i \land l_j \]
Experimental results for extraction of circles and lines
Experimental results for extraction of circles and lines.
Representation of information for detection of bilateral symmetry

For each pair of points:

For each pair of lines:

And the perceptual saliency of a symmetry axis is:

\[
W(L, D) = \sum_i \exp \left[ \frac{2}{\sigma^2} \frac{(L p_{ic} \cdot L) \cdot p_{jc}}{p_{ic}} \right]
\]

Where:

\( p_{jc} \) is the nearest pixel to the point \( L p_i L \).

\( \sigma \) is a parameter that fix the size of the neighbourhood in which we look for point \( p_{jc} \).
And the perceptual saliency of a symmetry axis is:

$$L_{\theta} = R \left( \frac{\theta}{2} \right) l_i \tilde{R} \left( \frac{\theta}{2} \right),$$

$$\theta_1 = \text{atan} \left( \frac{|l_i \wedge l_j|}{|l_i \cdot l_j|} \right), \quad \theta_2 = \theta_1 + \pi.$$  

- And the perceptual saliency of a symmetry axis is:

$$W(L, D) = \sum_i \exp \left[ \frac{2 \left( L p_{ic} \tilde{L} \right) \cdot p_{jc}}{\sigma^2} \right]$$

Where:

- $p_{jc}$ is the nearest pixel to the point $Lp_i L$.
- $\sigma$ is a parameter that fix the size of the neighbourhood in which we look for point $p_{jc}$. 

For each pair of intersection lines:

For each pair of parallel lines:

$$L_d = T \left( \frac{d}{2} \right) l_i \tilde{T} \left( \frac{d}{2} \right), \quad d = |(l_i \wedge l_j) \cdot n_i|.$$
Voting process for extraction of symmetry axis
Experimental results for extraction of symmetry axis
Geometric Computing for Cybernetics

edb@gdl.cinvestav.mx

http://www.gdl.cinvestav.mx