1) “Geometry of the squared distance function to curves and surfaces,” Pottmann et. al, and

2) “Registration without ICP,” Pottmann et. al

Summaries by Joyoni Dey,
Oct 22, 2003

1) Geometry of Distance function…

Planar Case: Squared distance function of points in a plane to a curve in the plane

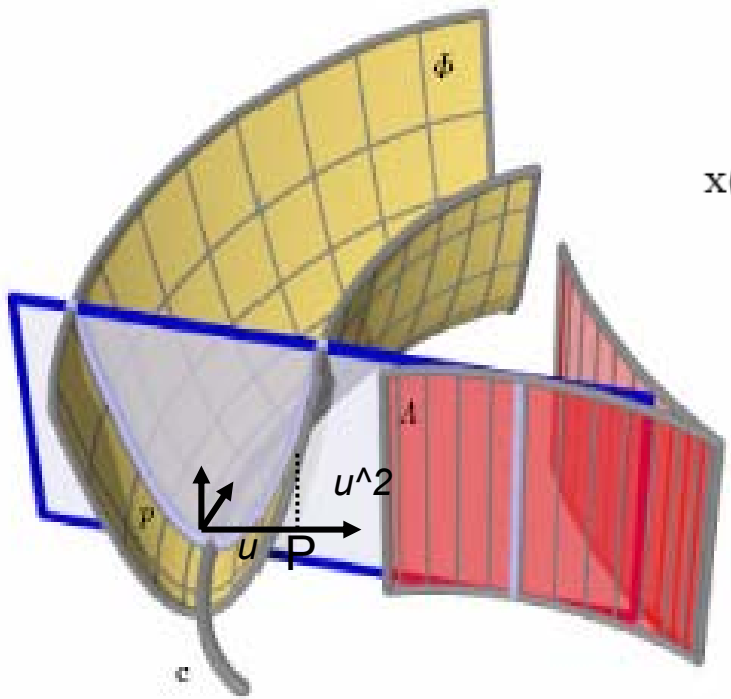


Fig. 1. When the normal plane of the planar curve c rolls on the evolute cylinder A of c, the parabola p generates the moulding surface Φ

Normal plane cuts a parabola on "Moulding" Surface Φ

$$\mathbf{x}(t,u) = \mathbf{c}(t) + u\mathbf{e}_2(t) + u^2\mathbf{e}_3(t). \qquad (1)$$

Frenet system instantenously rotates about the axis of "osculating circle".
Normal plane rolls along "evolute" Cylinder Λ with rulings parallel to z-axis

Another example of evolution of Surface

Vertical line may intersect surface at several points:
1) Self-intersections need to be trimmed,
2) Multi-valued cases, smallest height points retained (visibility test)

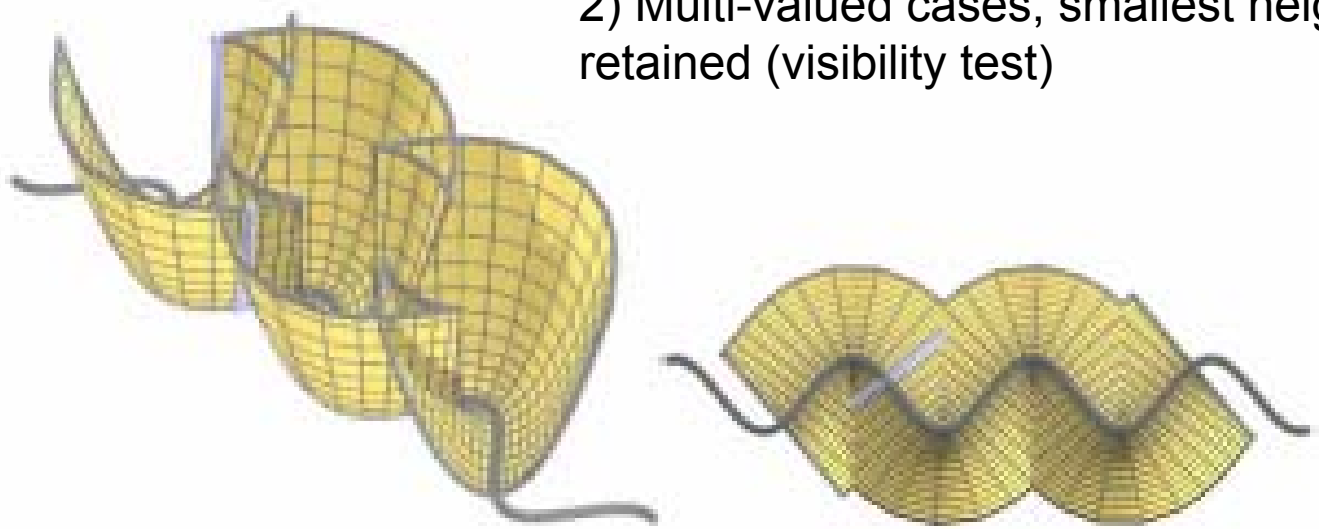
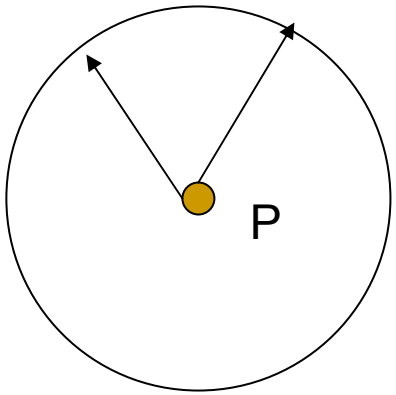


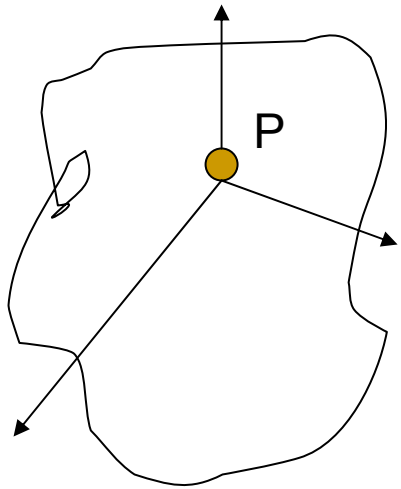
Fig. 2. A moulding surface Φ with self-intersections in an axonometric view (**left**) and viewed from below, where the visible points correspond to the graph Γ of the squared distance function to a sine curve (**right**)

Trimming Examples

Self-intersection and Multi-valued



If the curve is a circle, all points equidistant to center, P. Distance surface will self-intersect at P



Another curve where several curve normals can pass through Point P; take smallest distance

Blue Quadratic Surface has second order contact with Moulding Surface

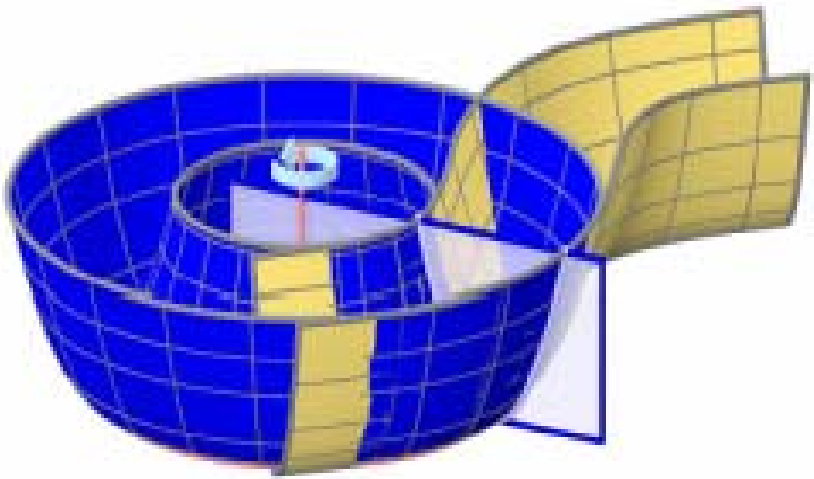
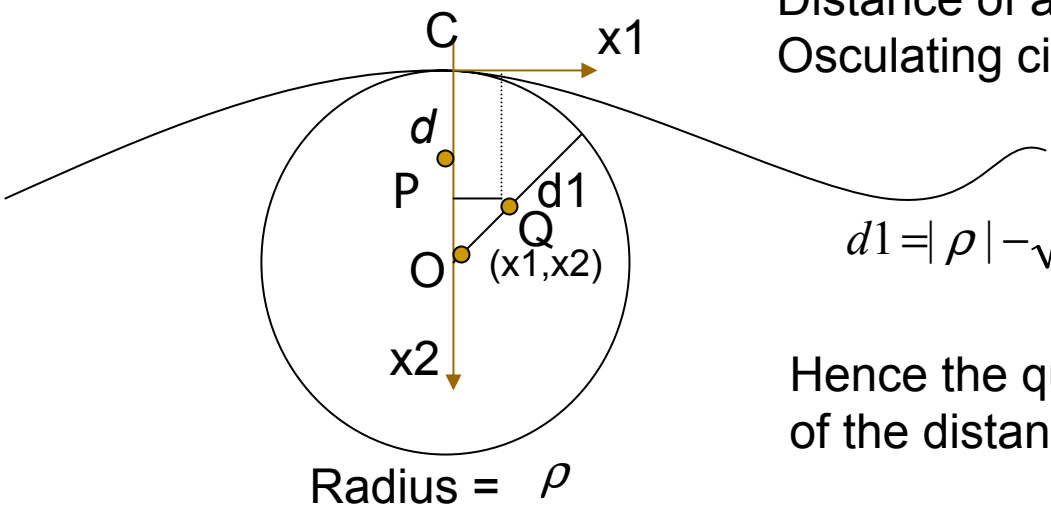


Fig. 4. Surface of revolution Ψ which has second order contact with the moulding surface Φ at all points of $p(t_0)$

Local quadratic approximation of distance to curve: distance to the osculating circle



Distance of a local point Q (x1,x2) to the Osculating circle given by d1

$$d1 = |\rho| - \sqrt{x_1\text{^}2 + (x_2 - \rho)\text{^}2}$$

Hence the quadratic approximation of the distance function is:

$$f(x_1, x_2) = \left(\sqrt{x_1^2 + (x_2 - \varrho)^2} - |\varrho| \right)^2 . \qquad (2)$$

Taylor Series expansion around P

$$F_d(x_1, x_2) = \frac{d}{d - \varrho} x_1^2 + x_2^2. \qquad (3)$$

– For $d = 0$ we get the Taylor approximant $F_0 = x_2^2$ at the normal footpoint.

For $d \approx 0$ second order approximation of distance function is just distance to the tangent plane (dotted line in diagram of previous slide)

– For $d \to \infty$, the Taylor approximant tends to $F_\infty = x_1^2 + x_2^2$. This is the squared distance function to the footpoint $\mathbf{c}(t_0)$. The graph Γ_∞ of F_∞ is a paraboloid of revolution.

– For general d, it may be advantageous to view F as combination of F_0 and F_∞,

$$F_d(x_1, x_2) = \frac{d}{d - \varrho} \left(x_1^2 + x_2^2 \right) - \frac{\varrho}{d - \varrho} x_2^2 = \frac{d}{d - \varrho} F_\infty - \frac{\varrho}{d - \varrho} F_0. \qquad (4)$$

Quadratic Approx to Distance function for Surfaces

$$F_d(x_1, x_2, x_3) = \frac{d}{d - \varrho_1} x_1^2 + \frac{d}{d - \varrho_2} x_2^2 + x_3^2. \qquad (7)$$

Again for $d \approx 0$ second order approximation of distance function is just distance to the tangent plane

2) Registration without ICP

- Registering a CAD surface model to a cloud of points x_i
- ICP Algorithm:

  - Find closest corresponding points y_i

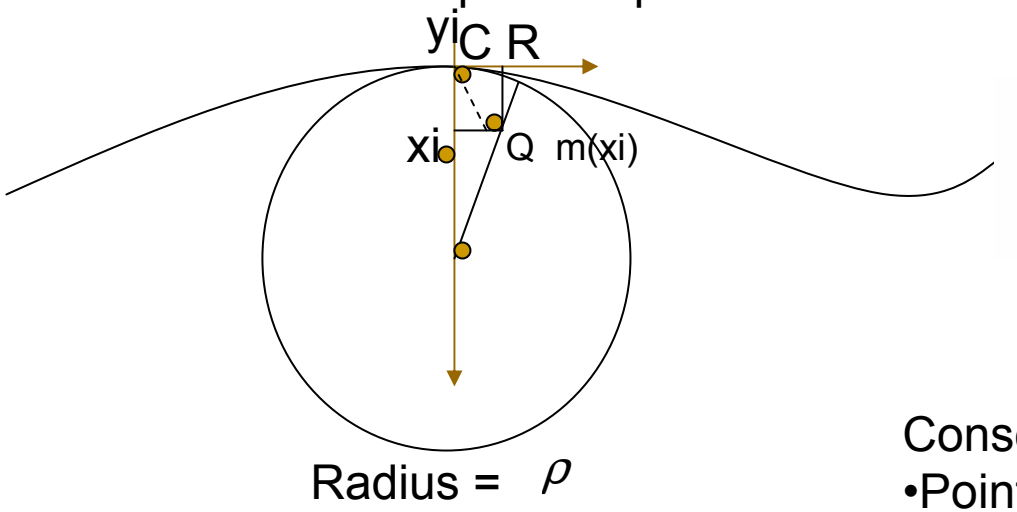
  - Transform Rigidly (rotation + translation) by m and minimize squared distance:

$$F = \sum_{i=1}^{N} \|m(\mathbf{x}_i) - \mathbf{y}_i\|^2. \qquad (1)$$

Note: Minimization can be done explicitly

Flawed for close points!

Okay when points are far away…
But when points are close to the surface, ICP takes point-to-point distance CQ when one should take point-to-plane distance RQ.



$$F = \sum_{i=1}^{N} \|m(\mathbf{x}_i) - \mathbf{y}_i\|^2.$$

Consequences:
•Points will tend to cluster around "footpoints" yi
•convergence along tangent plane will be slow

First Linearize "m" as much as possible

The rigid-body transform is broken into a helical motion and then a very small rotation + translation. Optimization done only with respect to helical motion.

$$\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}.$$

Helical motion **xi** → **xi**+v(**xi**)

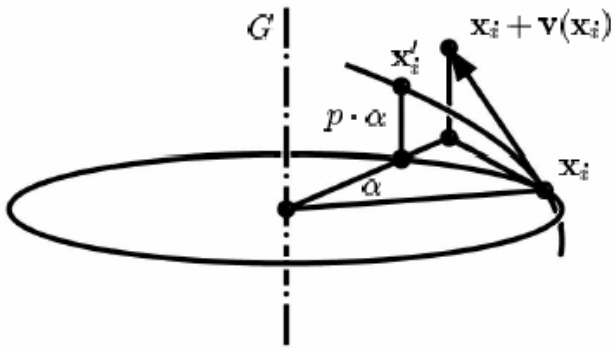


FIG. 4 New position $\mathbf{x}_i'$ of a point $\mathbf{x}_i$.

Rotation+translation **xi**+v(**xi**) → **xi**′

$$\mathbf{g} = \frac{\mathbf{c}}{\|\mathbf{c}\|}, \qquad \bar{\mathbf{g}} = \frac{\bar{\mathbf{c}} - p\mathbf{c}}{\|\mathbf{c}\|}, \qquad p = \frac{\mathbf{c} \cdot \bar{\mathbf{c}}}{\mathbf{c}^2}, \qquad \omega = \|\mathbf{c}\|.$$

a rotation about this axis G through an angle of $\alpha = \arctan \|\mathbf{c}\|$

a translation parallel to G by the distance $p \cdot \alpha$ (see Fig. 4).

Modification to ICP

Step 1: Find yi, closest points to xi on surface

Step 2: Minimize *distance to tangent plane*

wrt to "linear" part of the transformation –

find the $\bar{\mathbf{c}}$ and **c** (6 parameters again). Explicit again.

$$\mathbf{x}_i \mapsto \mathbf{x}_i + \mathbf{V}(\mathbf{x}_i)$$

Minimize

$$F(C) := F(\mathbf{c}, \bar{\mathbf{c}}) = \sum_i (d_i + \mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i))^2, \qquad (17)$$

Step 3: Do the small final twist to get the true rotation and translation.

Replace old point cloud with rotated point cloud

and iterate…

Caveats

Note 1: Last step is a *small* twist

Using the underlying helical motion is furthermore justified by the fact that for $\mathbf{x}_i$ we do not know the exact corresponding point on the surface anyway, we are moving the point closer to the tangent plane, and we iterate the whole procedure to find the optimal match.

Note 2: Degeneracy exists for certain surfaces, such as planes, general cylinders,…,

surfaces of revolution, and helical surfaces (**cf. [9, 11]**). These surfaces are characterized by the fact that there exists a vector field $\mathbf{v}(\mathbf{x}) = \mathbf{c} + \bar{\mathbf{c}} \times \mathbf{x}$ such that for each surface point $\mathbf{p}$ the vector $\mathbf{v}(\mathbf{p})$ is tangential to the surface in $\mathbf{p}$.

In the context of registration, these kinematic surfaces play a special role as well. After the registration of a point cloud to such a surface, the point cloud can still be moved tangentially to the surface without increasing the objective function $F(C)$ in Equ. (19). Thus, in the special case of kinematic surfaces the linear system (20) gets ill-conditioned. Whereas the standard ICP algorithm heavily punishes tangential movement (which slows the convergence behavior), minimizing $F(C)$ in Equ. (19) does not restrict tangential movement at all.

It is straightforward to combine the functional $F(C)$ with a functional

$$F'(C) := \sum_i (\mathbf{x}_i - \mathbf{y}_i + \bar{\mathbf{c}} + \mathbf{c} \times \bar{\mathbf{x}}_i)^2,$$

which describes the sum of squared distances of the points $\mathbf{x}_i + \mathbf{v}(\mathbf{x}_i)$ to the normal footpoints $\mathbf{y}_i$. Minimizing the quadratic functional $\bar{F}(C) = F(C) + \omega F'(C)$, where ω is a small but positive weight, again leads to the solution of a linear system.

Results

Synthetic data, added Gaussian Noise

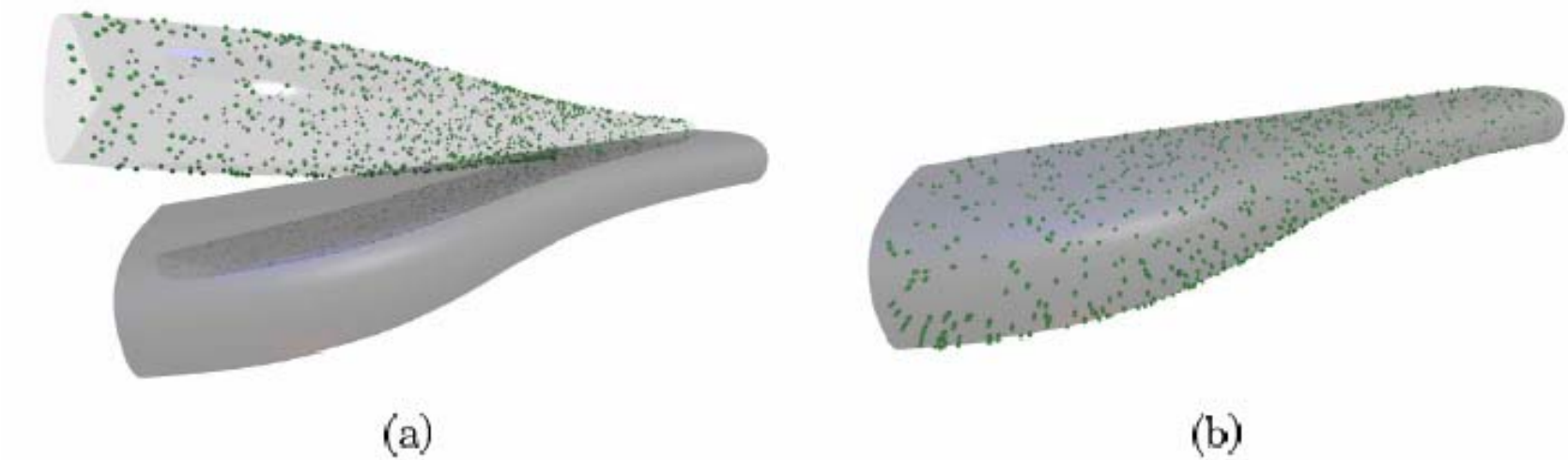


FIG. 5 Registration of a point cloud to a surface. Initial position (a) and final position after 4 iterations (b).

Comparison to ICP

When points are close to the surface, ICP has a residual error while point-to-plane algorithm performs much better.

One question I had: The final MS error seems zero for new algorithm. But some residual error still expected of the order of Gaussian noise variance ? Maybe the y-axis scale is too big to showing it.

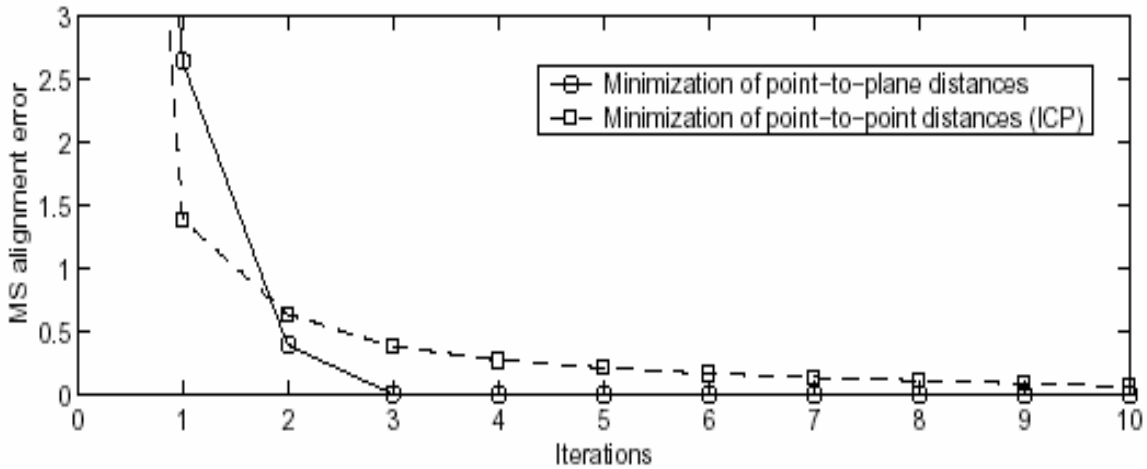


FIG. 6 Comparison of the convergence rate for minimizing point-to-plane distances vs. point-to-point distances (ICP).

Comment

- Felt that calling the algorithm non-ICP is not justified because “correspondence” is still obtained.

Future Directions prescribed by authors

- Instead of registration with a ridig body motion, we might allow a uniform scaling, i.e., a similarity. This is a minor change in the presented algorithm, since the velocity field is still linear and just has one more real parameter σ,

$$\mathbf{v}(\mathbf{x}) = \sigma\mathbf{x} + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}. \tag{21}$$

- In extension of [10], we have to investigate other quadratic approximants of the squared distance function to a surface. In particular, we need an efficient way of computing local quadratic approximants if the model shape is just given as a point cloud.
- Fast registration for industrial inspection requires the development of a spatial data structure, computed in a preprocessing step from the given model shape, such that the necessary quadratic approximants can be quickly computed or retrieved from that structure.
- It seems to be interesting to look at a hierarchical representation of the quadratic approximants of d^2, and use it efficiently in the various iteration steps of the registration procedure.
- The use of instantaneous kinematics allows us to extend the idea to the simultaneous registration of more than two geometric objects (partial scans).