Augmented Reality and Point Set Matching

Author: Karl - Walter Sillaots

What is Augmented Reality (AR)?

- Mixed interactive experience – real-world environment with computer-generated information
  - Visual
  - Auditory
  - Optionally, haptic, neural, smell
- System that fulfils 3 conditions:
  - A combination of real and virtual worlds
  - Real-time interaction
  - Accurate 3D registration of virtual and real objects

Current Use Cases?

- Not used often in commercial products
- Furniture
- Visual aid
- “Augments” the environment

Devices

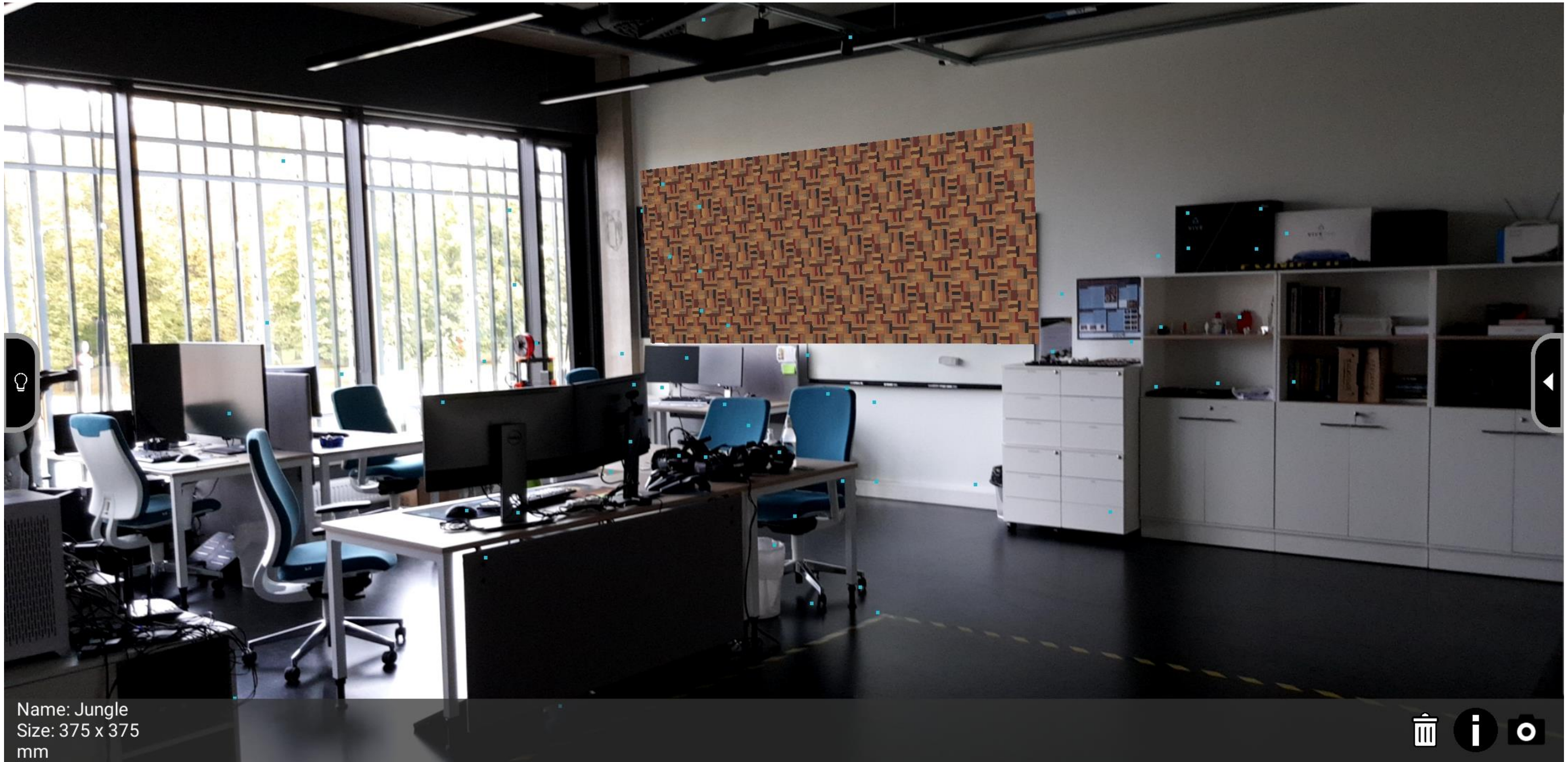
- Camera – to perceive the world and make decisions from it
  - A computer with a camera
  - Smart devices (phones, tablets)
  - Head-Mounted Display

Virtual Reality or Augmented Reality?

- VR – The perception of reality is all based on virtual information

Virtual Reality or Augmented Reality?

- AR – Part of the environment is "real", with virtual objects on top of it



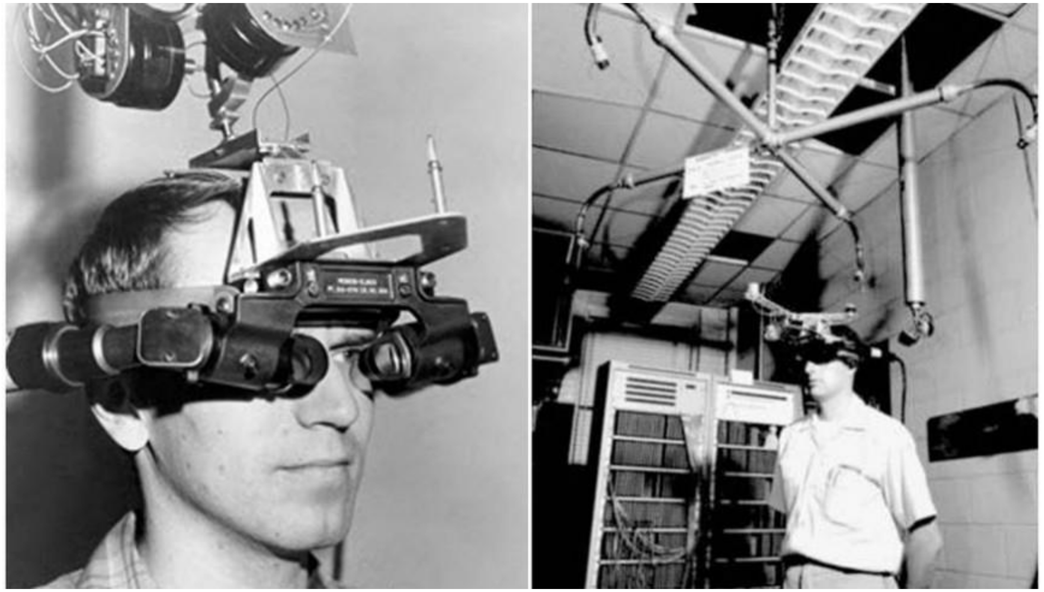
Augmented Reality History

1968

- Note - VR and AR didn’t exist at this point, only Artificial Reality

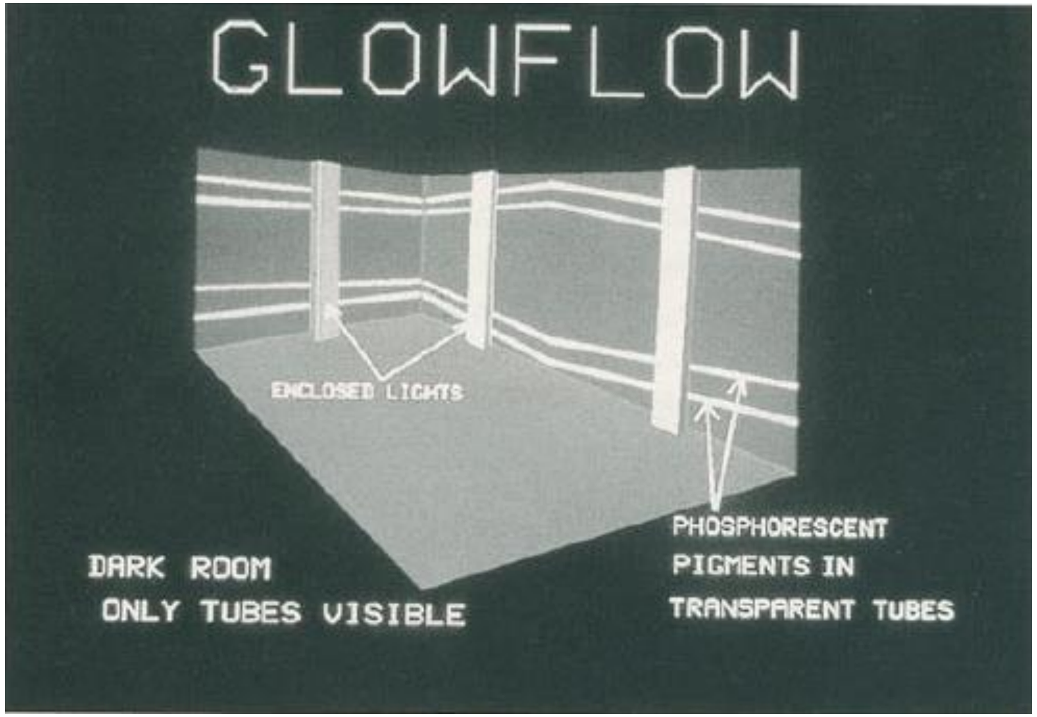
1968

- The Sword of Damocles
  - Created by Ivan Sutherland
- 2 CRT screens for either eye
  - Screens display 2D objects at different angles
- Head position sensors
- Wireframe objects



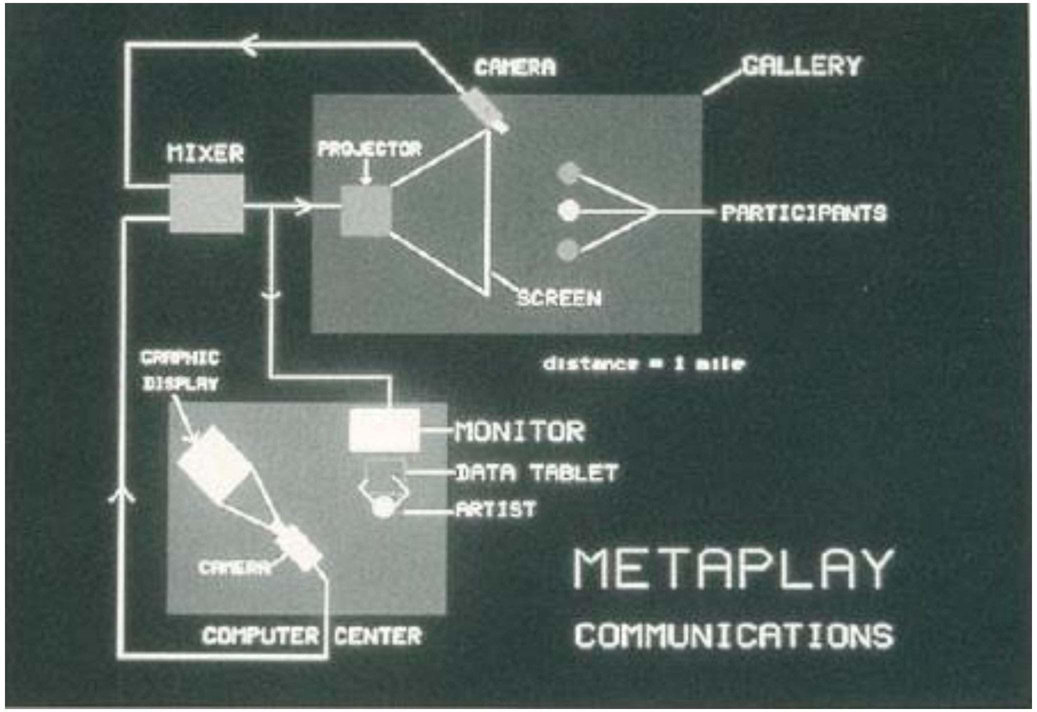
1969

- Glowflow
  - Myron Krueger
  - Wanted Artificial Reality that didn't need goggles or gloves to interact
- Dark room filled with phosphorescent pigments
- Floor sensors reacted to movement, synthesizer sounds, origin, lighting
- Issues with trigger mechanisms



1970

- Metaplay
- 2 Rooms connected through a video feed
- “Artist” sees others through camera, can interact with a drawing tablet
- Participants saw themselves project on media screen, on which the artist can write and draw on

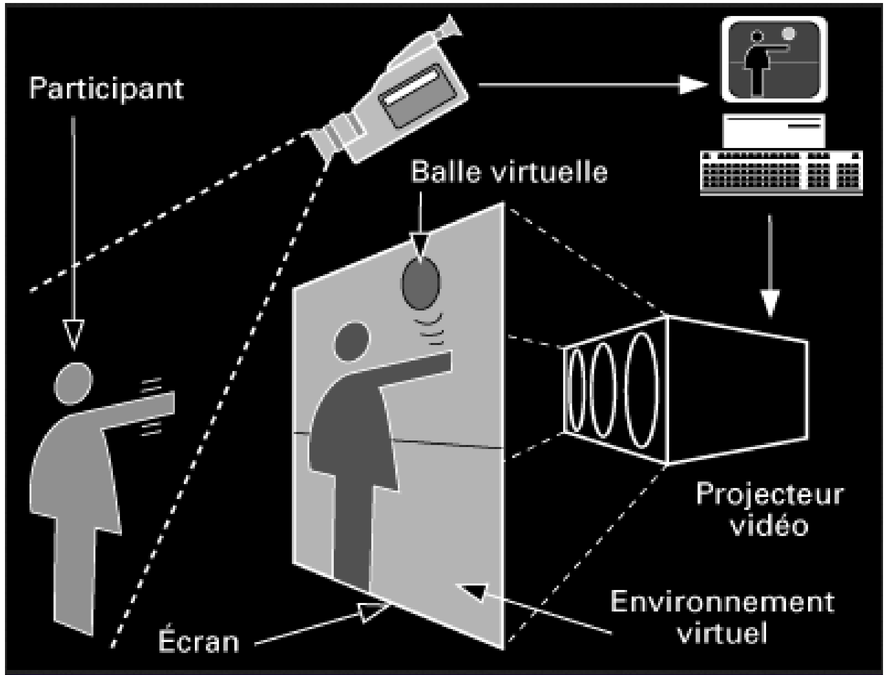


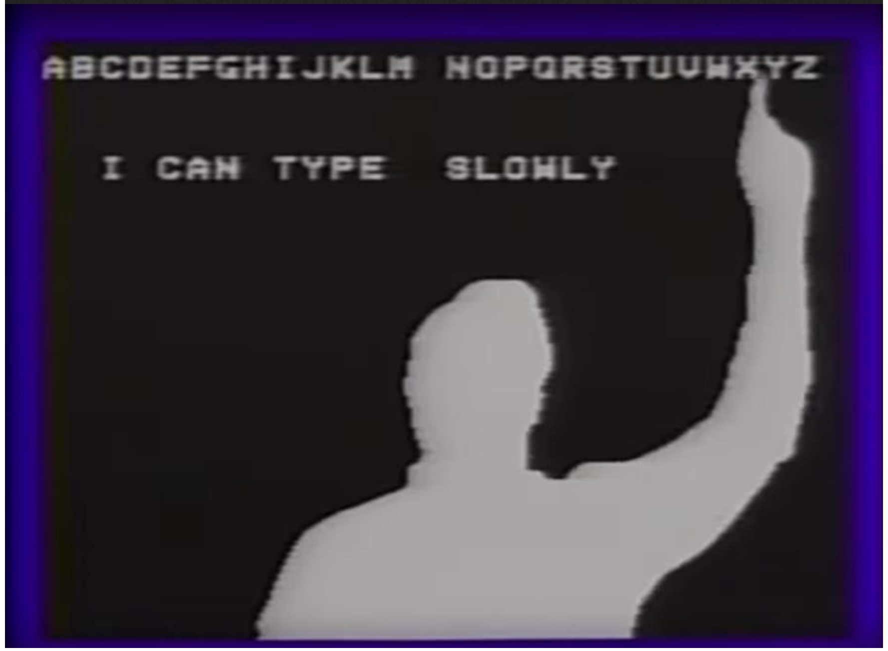
1971

- Psychic Space
- Advancement of Glowflow
- Improved sensor detection
- Invisible maze, user location displayed on screen with “wall” locations

1974

- Videoplace
- 2 rooms where image of one side projected to other. Both participants see the same image
- More interactivity
  - Playing with a virtual ball
  - Changing their displayed image, resizing, rotating
  - Typing text
- Computer Vision



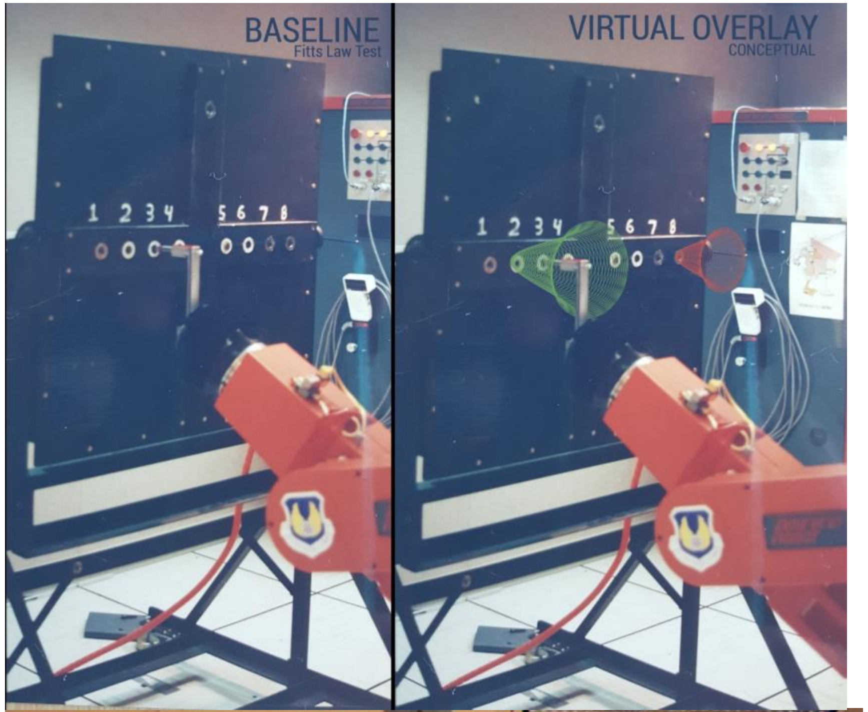


1990

- The term “Augmented Reality” was used for the first time
  - Thomas Caudell (David Mizell)
- Developed an industrial AR head mounted display
- Displays computer-generated diagrams of the manufacturing process
- Branched off as industrial AR (supporting industrial work)

1992

- Virtual Fixtures
  - Louis Rosenberg
- First fully functional AR system
- Displays and overlays virtual sensor information
- 3D graphics slow at that time, uses 2 physical robots instead
- Binocular magnifiers aligned user view with robot arms for better immersion



1994-1998

- Various entertainment use cases
- Sportsvision displaying a yellow line in American Football games
  - Individually displayed and updated for every camera showing games
  - Samples field / players for line occlusion effects

1999

- NASA hybrid synthetic vision system for spacecraft

2000

- Open-source software library ARToolKit developed.
- Video tracking to overlay virtual graphics on top of it.
- Still being updated today

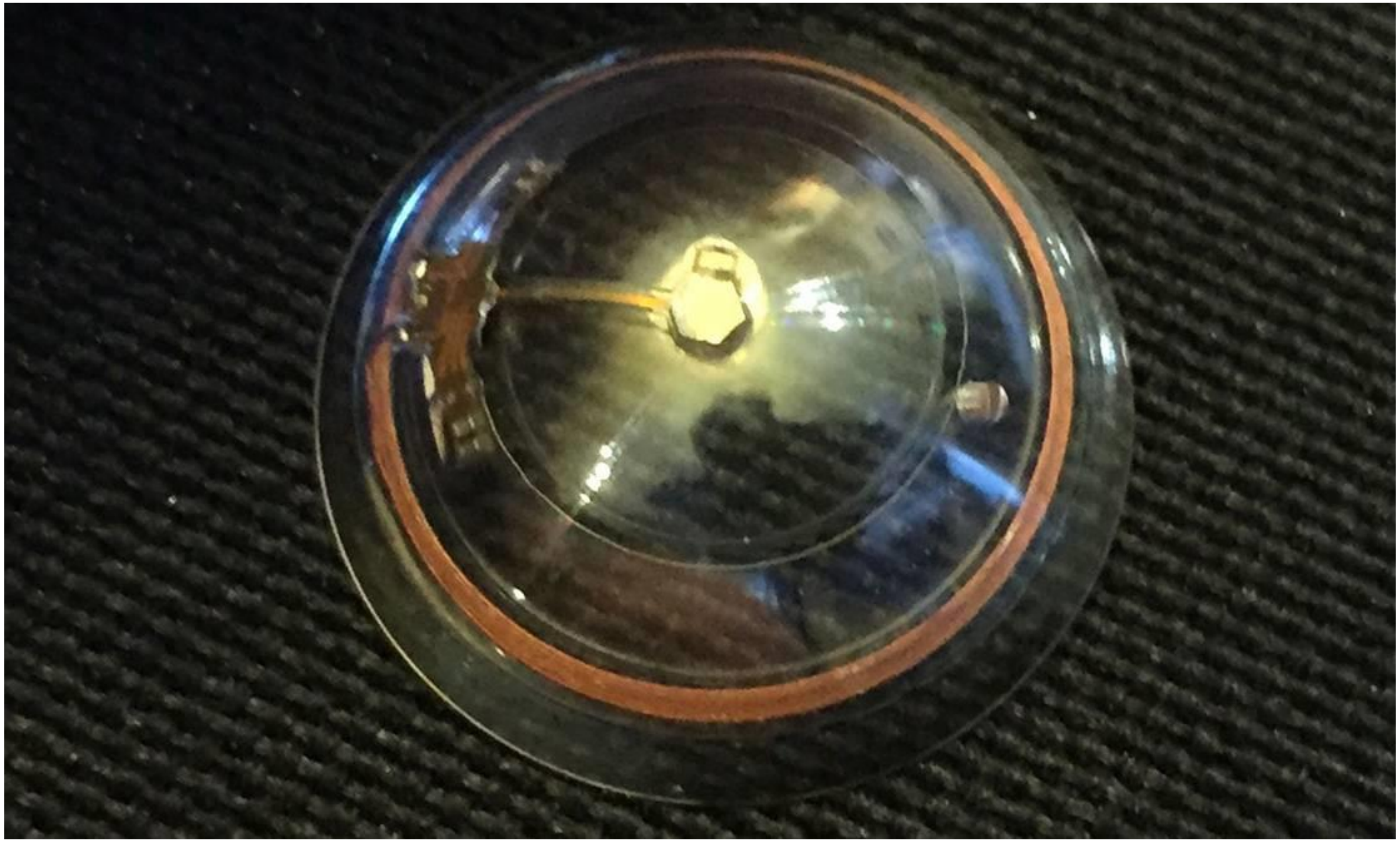
2001-2013

- 2003 – NFL usage of Skycam for virtual markers
- 2009 – Esquire Magazine with scannable barcodes to display AR content
- 2013 – Volkswagen using AR as car manuals (MARTA)

2014,2016

- Google Glass, Microsoft Hololens
- More immersive alternatives to smartphones
- Google Glass had a camera, touchpad, voice commands.
- Hololens have an accelerometer, gyroscope, magnetometer, depth-camera, multiple microphones, light sensor.

The Future?



Feature Detection

Feature Detection

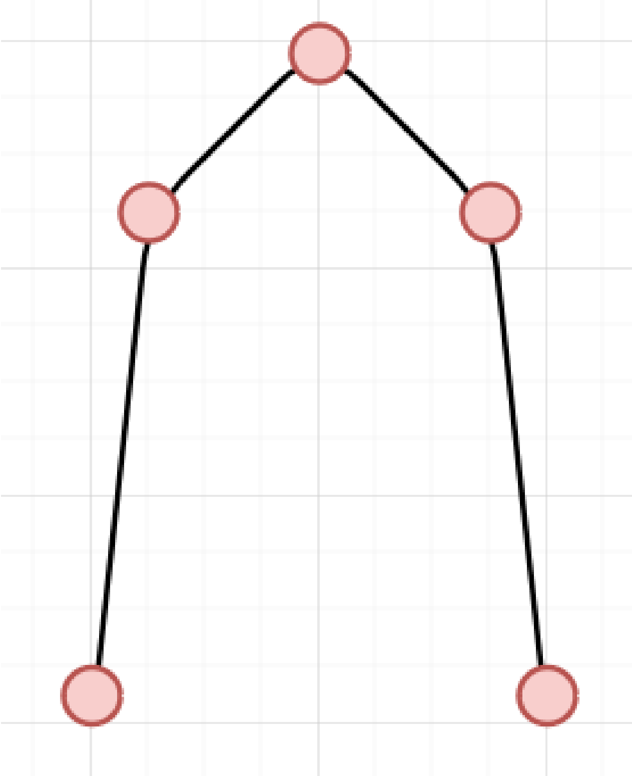
- A subcategory of computer vision and image processing
- Methods to compute image information
- “Feature” means “interesting” part of an image
- 4 general types
  - Edges
  - Corners (Interest points)
  - Blobs (Regions of interest points)
  - Ridges

Edge Detection

- Detect parts where brightness changes sharply
- Good in image processing, not AR.
- Marker-based solutions could still use this (searching for specific image)

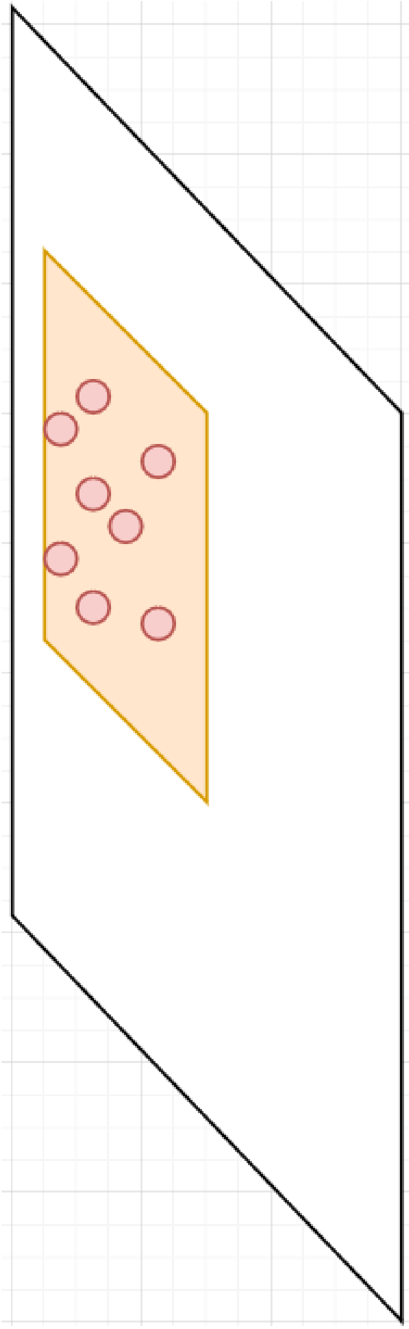
Corners

- An interest point where there is an intersection of 2 edges.
- Ends of a parabola
- Markerless AR



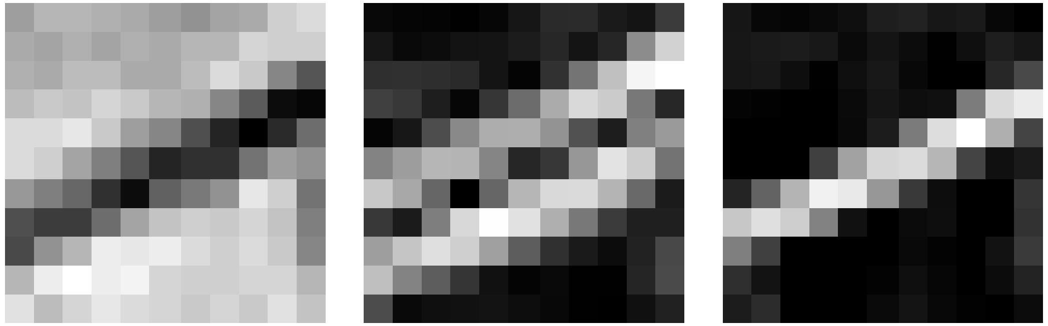
Blobs

- General analysis of image
- Find regions that differ in properties
  - Brightness
  - Color
- Smooth areas
- Markerless AR



Ridges

- For elongated objects
- 1D curves
- Harder to compute
- Road detection on aerial images



Thin line   Edge detector   Rdige filter

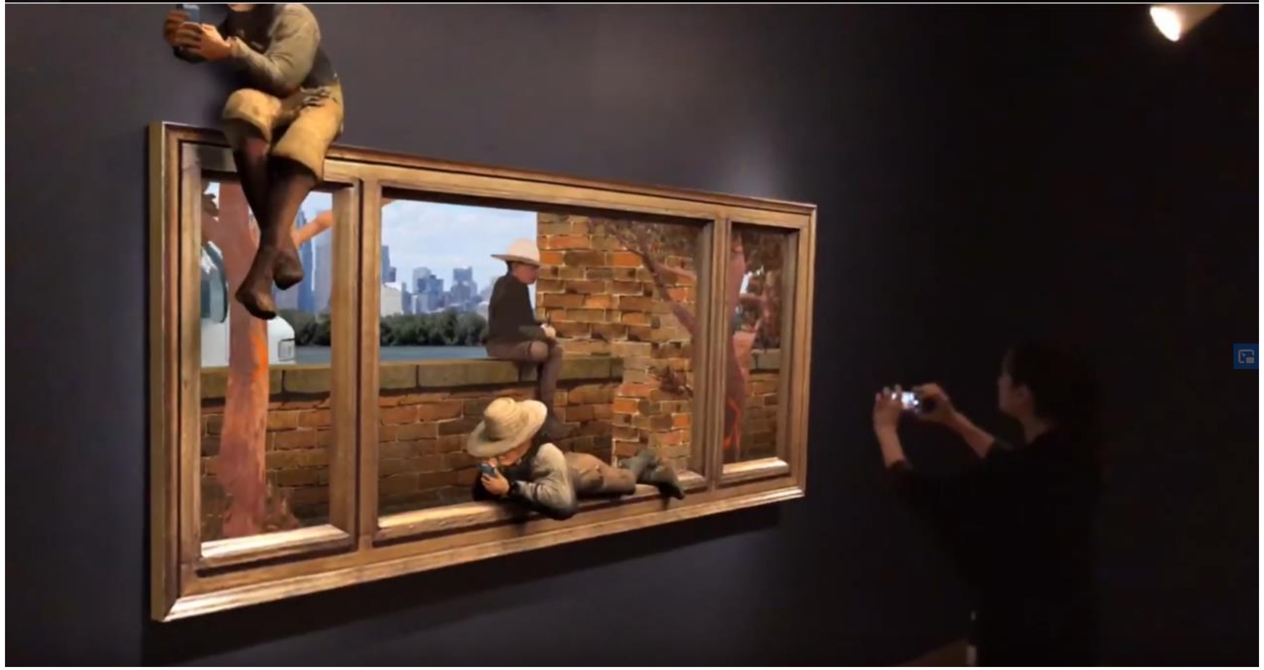
Object Placement

- Once the system understands the environment, it needs a way to place objects:
  - Marker-based
  - Markerless
  - Location based

Marker-Based

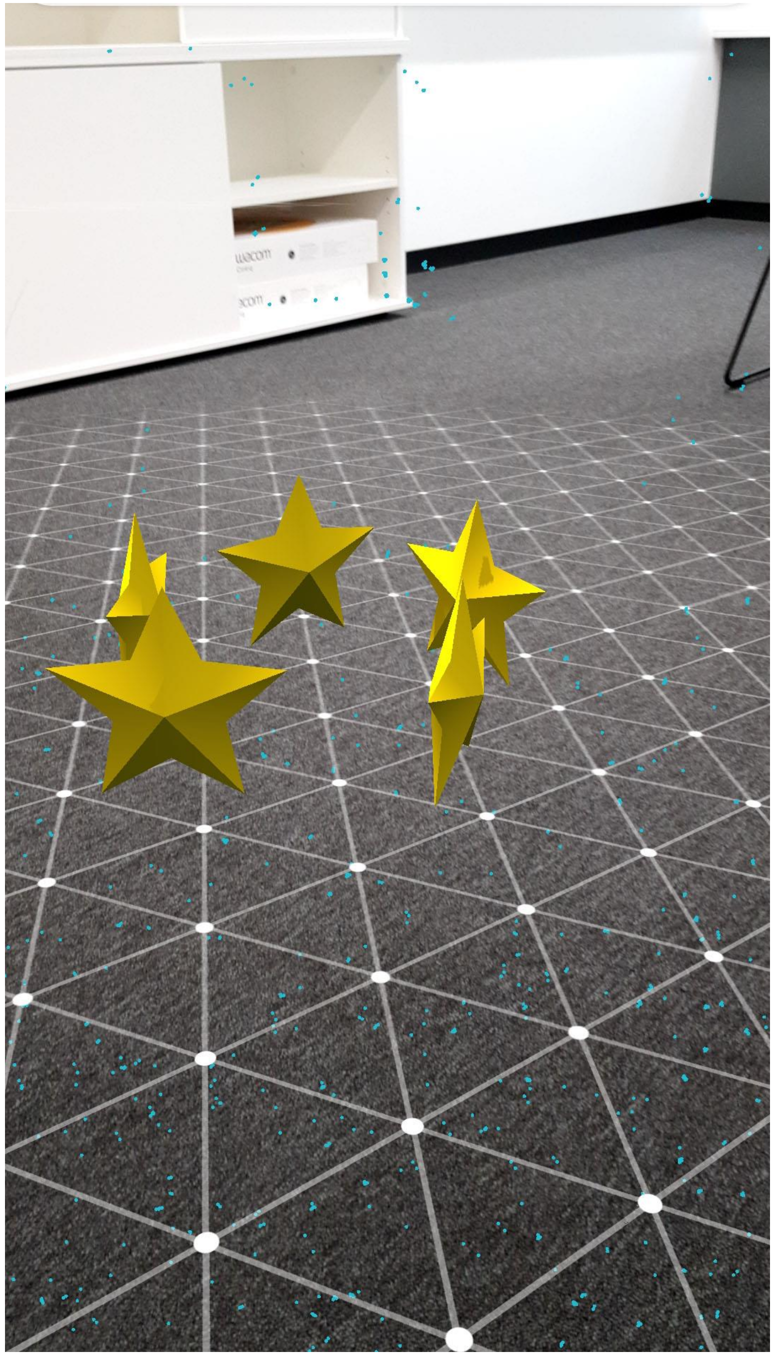
- Virtual object placed on a “marker”
  - Detectable image
- Can only visualize one object per image

Marker-Based



Markerless

- Uses feature points to detect surfaces
- Generates planes with feature points (plane detection)
- Objects placeable on planes.
- Only supports horizontal/vertical surfaces
- Doesn’t work with flat colored surfaces



Location based

- Uses real-world coordinates to estimate where to place objects
  - Latitude, longitude, altitude
- Better for outdoors environments
- More advanced features currently only on iOS

What about point cloud based?

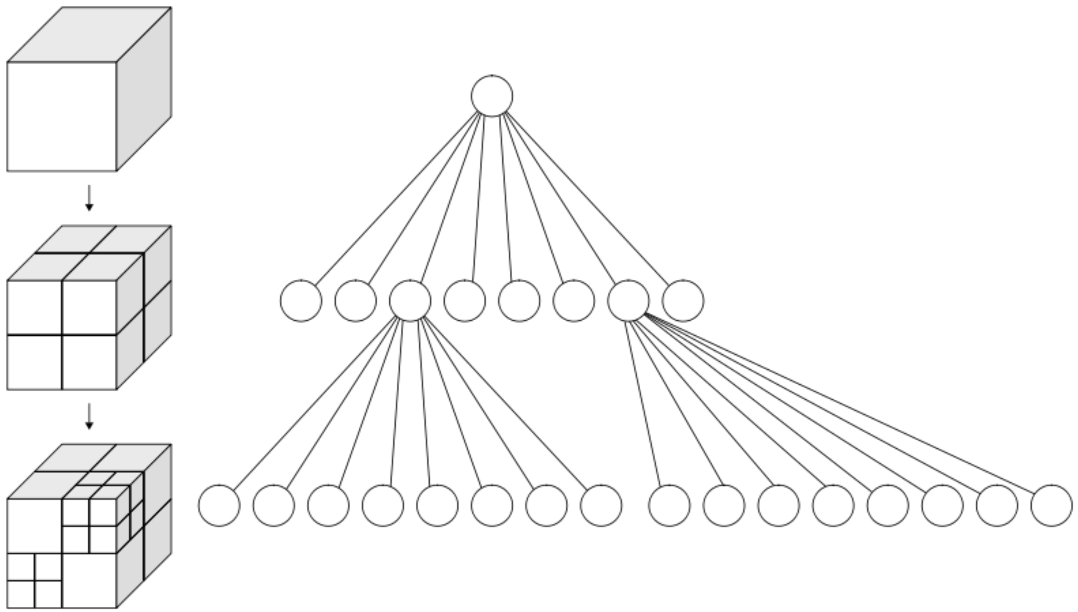
- Master thesis study – utilize AR on a lower level
- Detecting and saving feature points as point clouds
- Comparing point clouds and matching them
- Saved point cloud can have special object locations added to visualize once point clouds matched.

Problem 1 – Persistent Data

- Feature points are deleted when no longer seen by a device
- A separate container to store all visible points

Problem 2 – Excessive Data

- Too many points are saved, causes slowdown
- Check nearby points to see if a point should be added?
  - Would require traversing the entire collection of points for each point $O(n)$
- Octrees
  - 3D cube dividing to a minimum size
  - Only compare points within divided cube
  - Finding time now $O(\log n)$



Problem 3 – Saving Point Clouds

- A way to save and load point clouds in some form of data
- Each point is a 3D position
- Write the coordinates as a binary stream

Problem 4 – Comparing Point Clouds

- Many potential issues
  - Individual points, not planes – lots of comparisons between points
  - Tens of thousands of points per PC
  - Unique number of points per PC
  - Many points that can’t be directly matched with each other
  - Inaccurately placed points (outliers)
  - Coordinate systems of points clouds not matching

Point Set Registration

- Finding a spatial transformation that aligns two point clouds
  - Scaling
  - Rotation
  - Translation
- 2 finite point sets in a finite-dimensional real vector space

Point Set Registration

- Finding a spatial transformation that aligns two point clouds
  - Scaling
  - Rotation
  - Translation
- 2 finite point sets in a finite-dimensional real vector space
- Transformation types:
  - Rigid Registration – 2 separate point clouds matched without the distance between 2 points of a point cloud changing. Just translation and rotation of the point clouds
  - Non-Rigid Registration – allows non-linear transformation, scaling included

Point Set Registration

- One of the issues was point having deviations (outliers, different point locations)
- Non-rigid should be the solution
- Algorithms that cover this area:
  - SG4PCS
  - 2PNS
  - ACPD

Before That

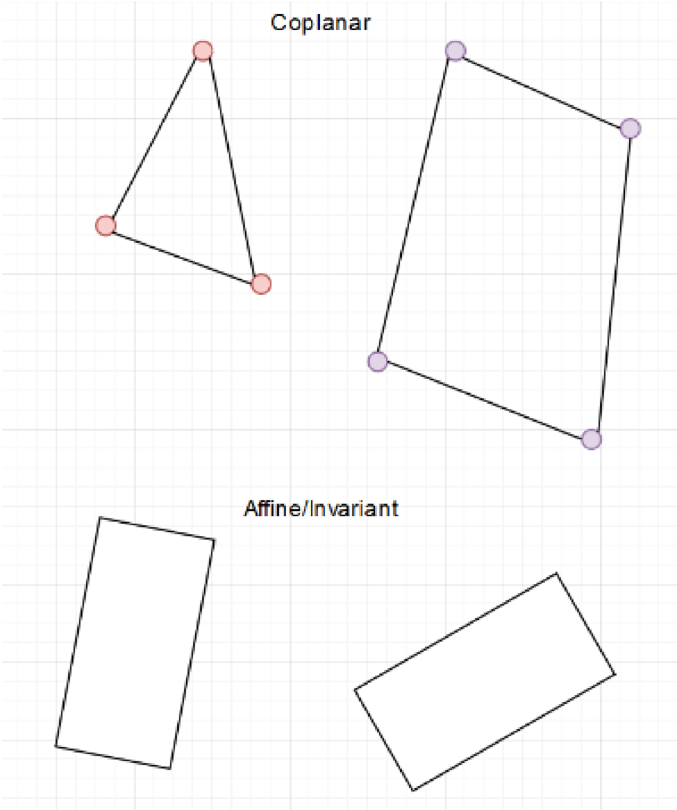
- RANSAC – Random Sample Consensus.
- Picks 3 random points from either point cloud
- Computations to count points from one point cloud that are close to points in other.
- If point count large enough, accepted as answer.
  - Otherwise it repeats.

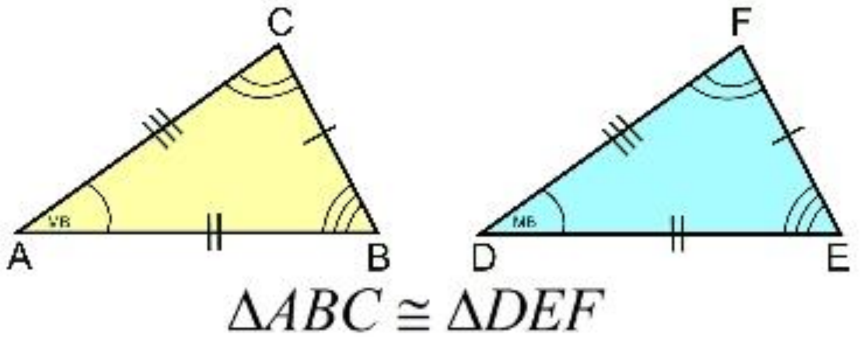
4-Point Congruent Sets (4PCS)

- “Fast”, robust alignment scheme for 3D point sets.
- Resilient to noise and outliers, even with small overlap

4-Point Congruent Sets (4PCS)

- 1. Uses coplanar sets of 4 point rather than minimum of 3, to apply a technique to match pairs of affine invariant ratios in 3D
  - Coplanar – Same plane
  - Affine – Preservation of lines and parallelism, but not distance
  - Invariant – Property that remains unchanged after transformations





4-Point Congruent Sets (4PCS)

- 2. Select a base of 4 coplanar points in PC P
- Find all 4-point sets in target PC Q, that are approximately congruent with base points.
- Done in $O(n^2 + k)$ time
  - n – Number of points in Q
  - k – Number of 4-point sets.

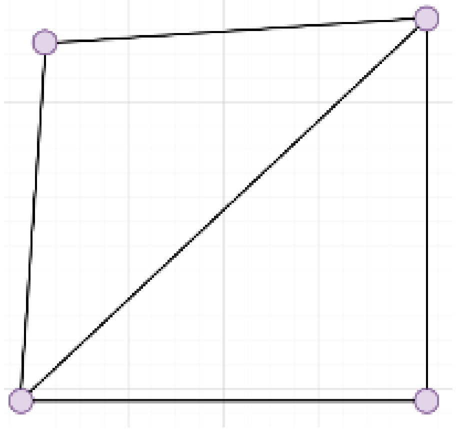
4-Point Congruent Sets (4PCS)

- 3. For each 4-point set from Q, compute aligning transformation T, retain best transformation based on Largest Common Pointset score.
- Repeat in RANSAC scheme until good solution found or maximum iterations reached.

- LCP problem
  - Given 2 point sets P and Q, under δ-congruence, Find largest countable subset of P called P' where distance between $T(P')$ and Q is less than δ. T being a rigid transform.

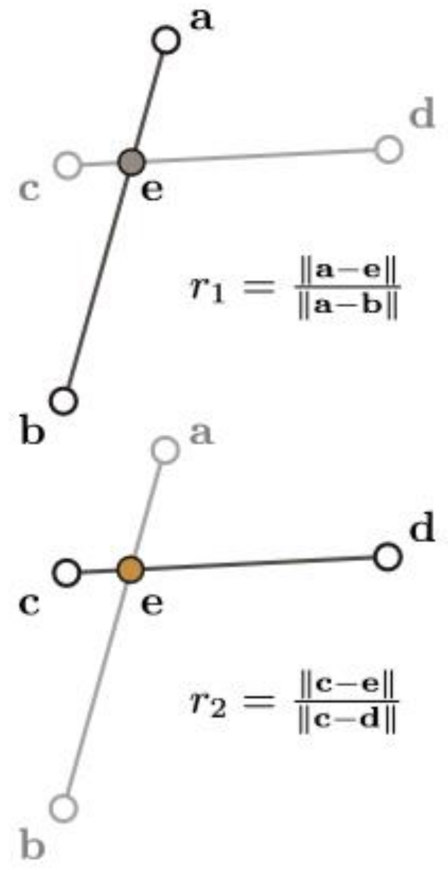
4-Point Congruent Sets (4PCS)

- 4. First step of each RANSAC iteration – pick a random base of 4 coplanar points.
- Picks first 3 randomly to create a wide triangle.
- 4^{th} selected is close to the planar of the other 3.
- Testing all S point in P, picking the one that fits best
- Complexity $O(S)$



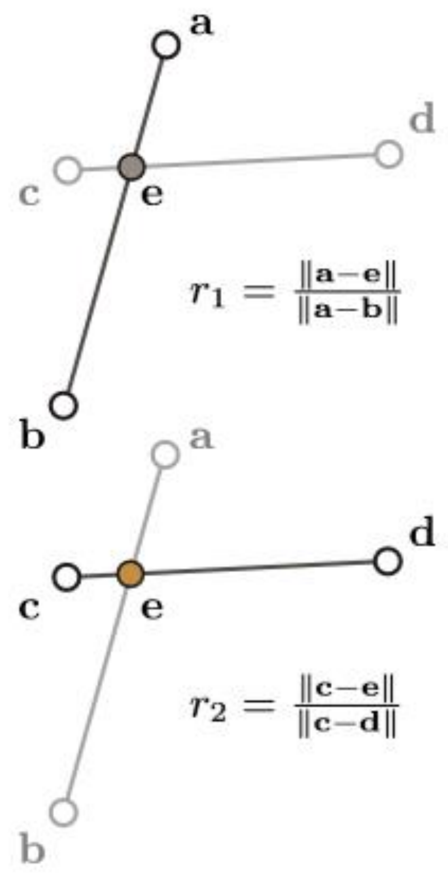
4-Point Congruent Sets (4PCS)

- 5. $\mathcal{B} = \{A, B, C, D\}$, where E is intersection of AB and CD
- $r_1 = \frac{||A-E||}{||A-B||}, r_2 = \frac{||C-E||}{||C-D||}$
- $d_1 = ||A - B||, d_2 = ||C - D||$
- Ratios r_1 and r_2 remain invariant under affine transformation, and therefore under rigid motion
- Distances preserved with rigid transformations – these 4 invariants used for searching congruent 4-point sets in Q



4-Point Congruent Sets (4PCS)

- $r_1 = \frac{||A-E||}{||A-B||}, r_2 = \frac{||C-E||}{||C-D||}$
- $d_1 = ||A - B||, d_2 = ||C - D||$
- 6. Extract all pairs of points and distance d_1 or d_2 from Q
- $O(N^2)$ time
  - N – amount of points in Q
- For each extracted pair $(Q_1, Q_2) \in Q$ with distances d_1 or d_2, compute intermediate points E_1, E_2

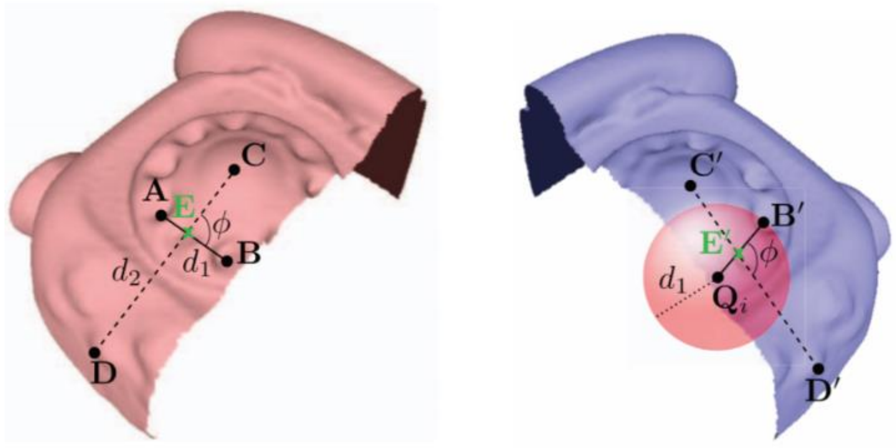


4-Point Congruent Sets (4PCS)

- For each extracted pair $(Q_1, Q_2) \in Q$ with distances d_1 or d_2, compute intermediate points E_1, E_2

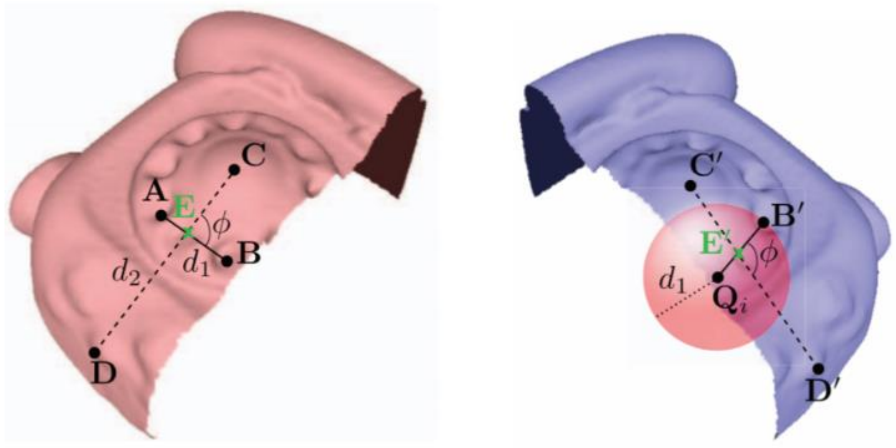
- 7. 2 pairs whose E_1, E_2 are coincident form a 4-point base related with $\mathcal{B}$ by an affine transformation

- Intermediate points from pairs at d_1 used to build an approximate range tree structure
  - $O(M \log M)$, where M is number of pairs
  - Query time $O(\log M + K)$, where K is number of points needed to get



4-Point Congruent Sets (4PCS)

- Intermediate points from pairs at d_1 used to build an approximate range tree structure
  - $O(M \log M)$, where M is number of pairs
  - Query time $O(\log M + K)$, where K is number of points needed to get
- 8. Intermediate points from pairs at d_2 used to query the tree
- Result is K 4-point sets from pairs
- $O(K)$ time to remove non-rigid sets



Super 4-Point Congruent Sets (S4PCS)

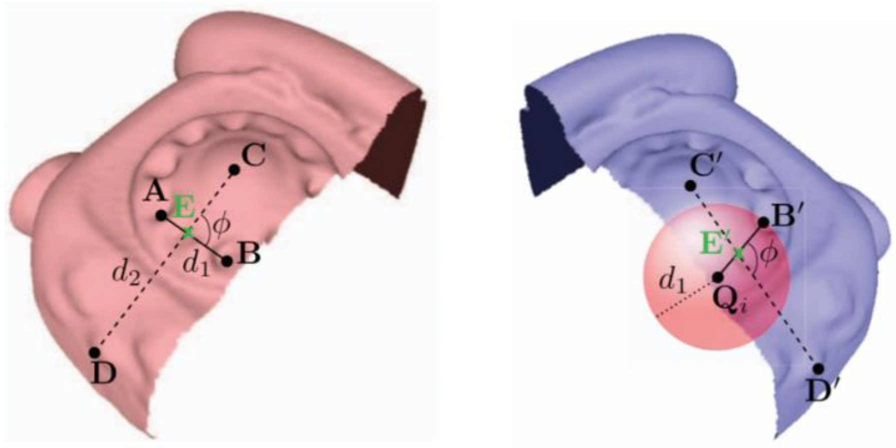
- Improves certain search stages to decrease complexity from quadratic to linear time.
- Supposedly works with about 25% overlap with 20% outlier margin.
- Complexity decreased to $O(N + M + K)$ by solving 2 bottlenecks
  - Pair Extraction (2) - $O(N^2 + K)$
  - Verification (8) - $O(K)$

Super 4-Point Congruent Sets (S4PCS)

- First bottleneck
- 2. Select a base of 4 coplanar points in PC P
- Find all 4-point sets in target PC Q, that are approximately congruent with base points.
- Done in $O(n^2 + k)$ time
  - n – Number of points in Q
  - k – Number of 4-point sets.

Super 4-Point Congruent Sets (S4PCS)

- 2. Select a base of 4 coplanar points in PC P
- Find all 4-point sets in target PC Q, that are approximately congruent with base points.
- Now find points close to spheres centered in $Q_i \in Q$ with radius $d_1 \pm \epsilon$ and $d_2 \pm \epsilon$, where ϵ is noise tolerance
- Pair extraction reduced to $O(n)$



Super 4-Point Congruent Sets (S4PCS)

- Second bottleneck
- Intermediate points from pairs at d_1 used to build an approximate range tree structure
  - $O(M \log M)$, where M is number of pairs
  - Query time $O(\log M + K)$, where K is number of points needed to get
- 8. Intermediate points from pairs at d_2 used to query the tree
- Result is K 4-point sets from pairs
- $O(K)$ time to remove non-rigid sets

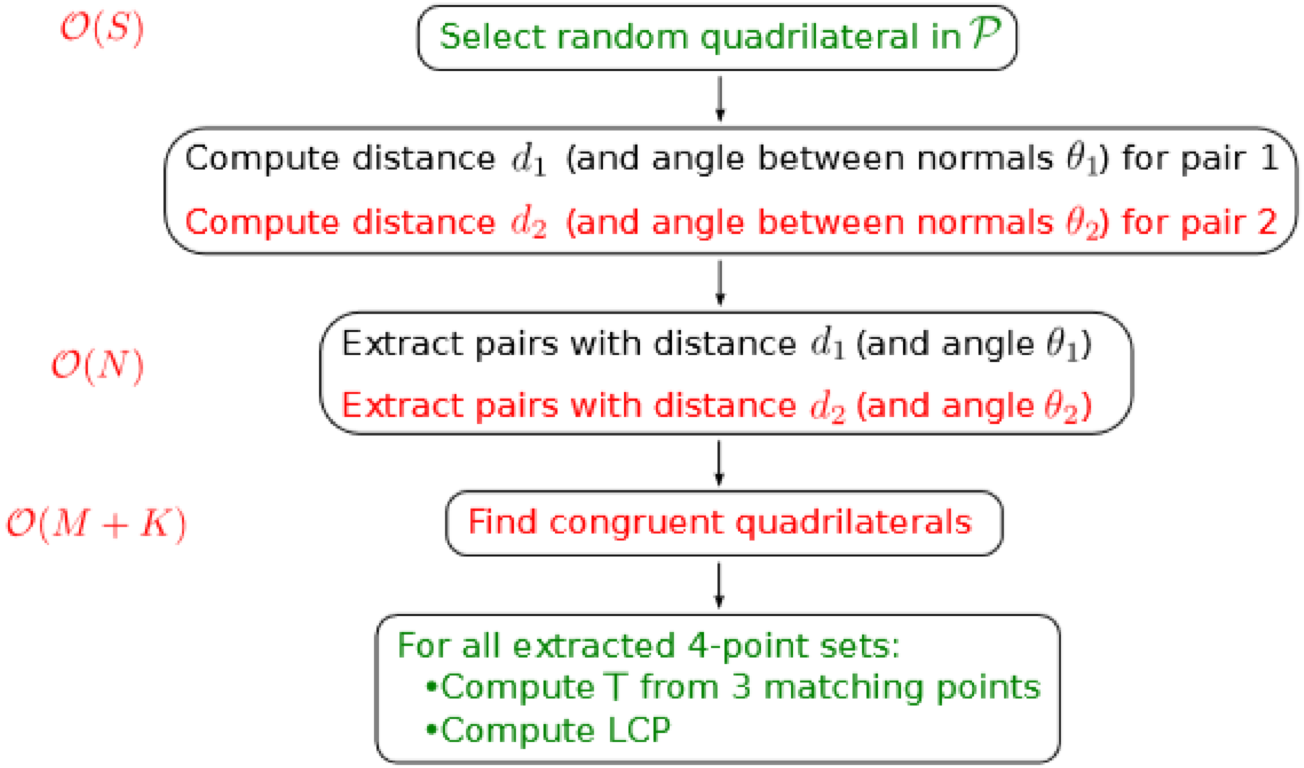
Super 4-Point Congruent Sets (S4PCS)

- $O(K)$ time to remove non-rigid sets
- Now extract only congruent 4-point bases that are rigid-invariant, so verification not needed.
- 4-point set congruent to base from P if it's composed of pairs with correct length (d_1, d_2) and angle ϕ between them similar to angle formed by the 2 pairs in the base

Super 4-Point Congruent Sets (S4PCS)

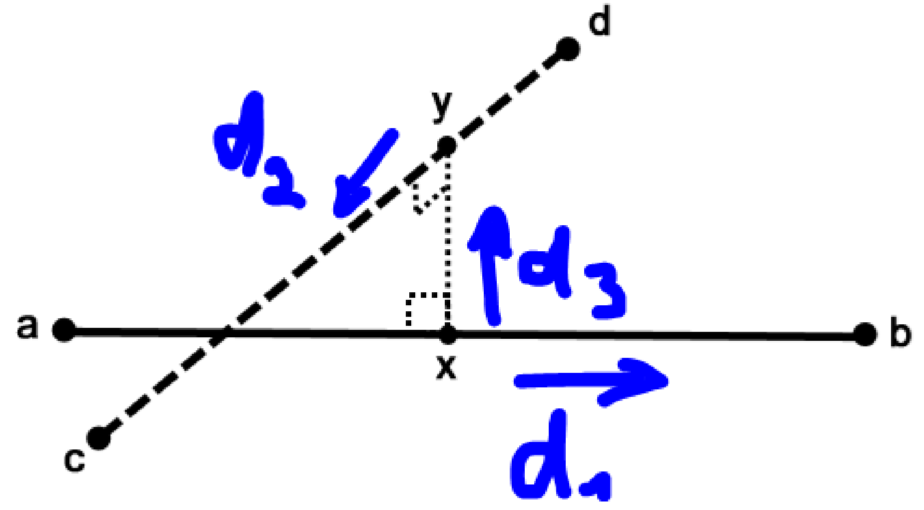
- Represent each point by intermediate point E and orientation.
- d_1 pairs hashed by this position and orientation, mapped to a spherical map.
- In query stage (7), get cells in a regular grid using E.
- Query sphere map using a d_2 pair direction, find all pairs with angle ϕ in regards to query direction. A cone of aperture 2ϕ is intersected around the query direction.
- Complexity $O(M + K)$

Super 4-Point Congruent Sets (S4PCS)



Generalized 4-Point Congruent Sets (G4PCS)

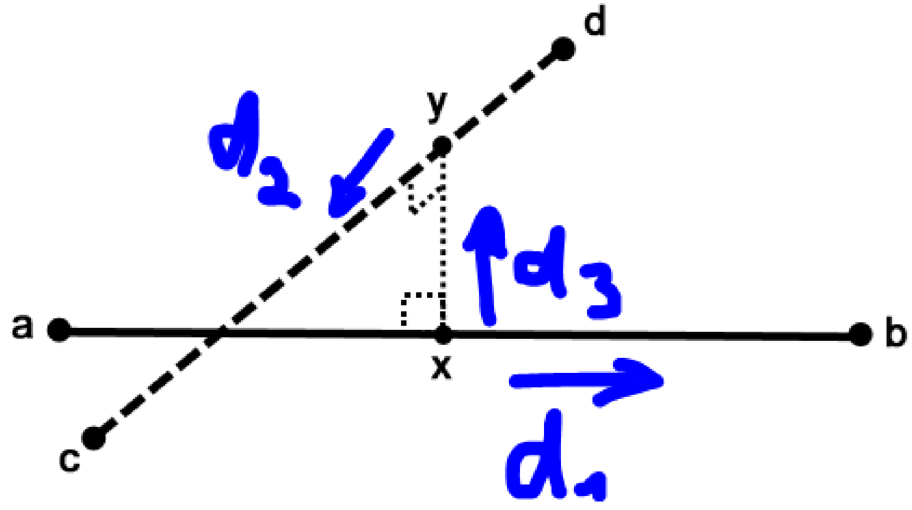
- Alternate advancement of 4PCS
- Different definition of the 4-point base
- $\mathrm{X} = \{A, B, C, D\}$, where AB does NOT always lie on same plane as CD



- $r_1 = \frac{||a-x||}{||a-b||}, r_2 = \frac{||c-y||}{||c-d||}$
- $d_3 = ||x - y||$

Generalized 4-Point Congruent Sets (G4PCS)

- Predefining values of d_1, d_2, d_3 to sample only bases that satisfy them.
- Any wide base now sampled from P, then d_i
- Bases storable in a 2D hash table based on ratios r_1, r_2
- Should lessen the amount of bases found



Super Generalized 4PCS (SG4PCS)

- Combination of S4PCS and G4PCS

Algorithm 1 The Super Generalized 4PCS Algorithm

Input: Target and source point sets, P and Q

Output: Best transformation according to LCP, T_{best}

```
d_1 = d_2 = fractional_overlap × model_diameter
Extract d_1, d_2 pairs from P
Extract d_1, d_2 pairs from Q
Initialize a 4D hash table H to store intersecting pairs
Compute all valid 3D intersections in Q and store in H
L = number of RANSAC iterations
T_best = 0
for l = 0 to L do
    B = random base from P
    r_1B, r_2B, d_3B, α_B are invariants of B
    C = ExtractCongruent(r_1B, r_2B, d_3B, α_B)
    T_B = Transformation with highest LCP from C
    if LCP(T_best) < LCP(T_B) then
        T_best = T_B
    end if
end for
```

2 Point Normal Sets (2PNS)

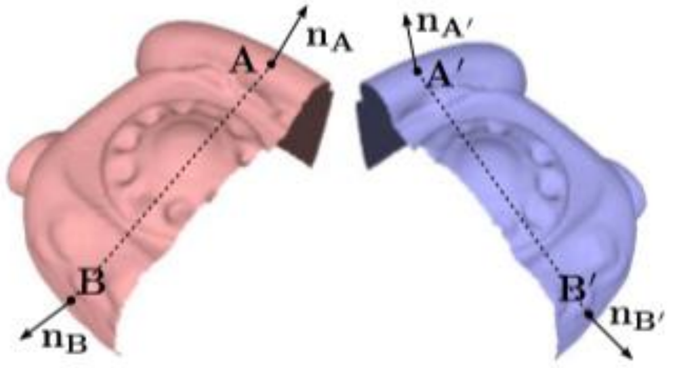
- Alternative to S4PCS using a different approach to 3D registration
- Using normals instead
- Rigid transformation T computable from 2 points plus normal of one point
- Reduces needed comparisons

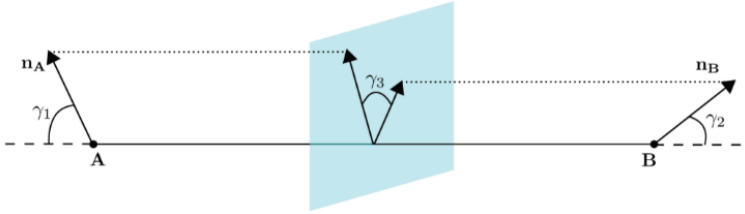
2 Point Normal Sets (2PNS)

- 1. Computing point normals
- PC surface normal estimation, PlanePCA?
- 2 solutions for each normal vector
- Fails when normals not estimated correctly
  - Sparse PC
  - Mostly sharp edges and corners

2 Point Normal Sets (2PNS)

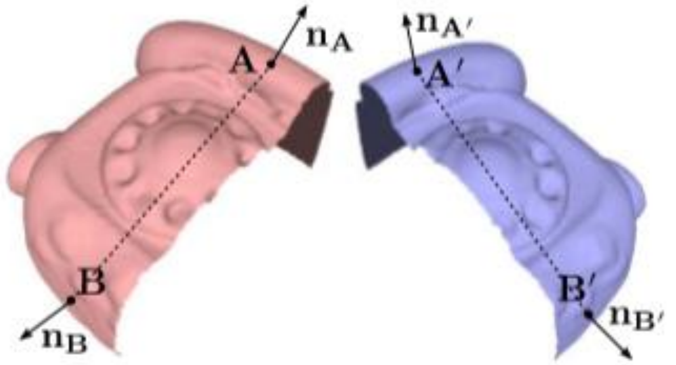
- 2. 2PNS search to obtain existing matches
- Take 2 points and normals from source PC P
- Extracts pairs
  - $d = ||A - B||$
  - $angle\ \theta = \angle(n_A, n_B)$
- Verify 3 additional angles to prevent non-rigid solutions
- Angles preserved under rigid transformation

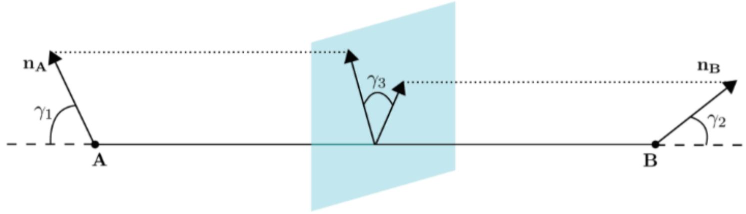




2 Point Normal Sets (2PNS)

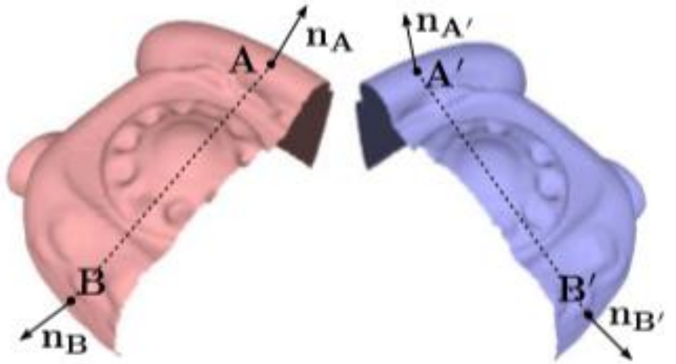
- 3. Estimation of R
- Let (A', B') with normals n_A, n_B, be pair of points in PC Q, congruent with pair in P
- Estimate rigid transformation and compute their rotation $R = \mathrm{R}_\alpha \cdot R_\beta$
  - R_a aligns vectors
  - R_β aligns normal vectors

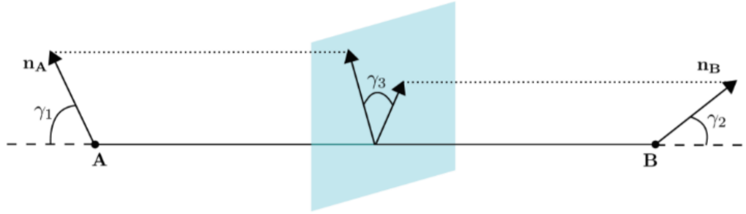




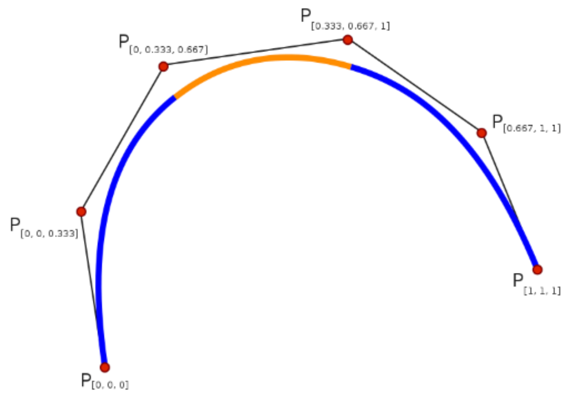
2 Point Normal Sets (2PNS)

- Estimate rigid transformation and compute their rotation $R = \mathrm{R}_\alpha \cdot R_\beta$
  - R_α aligns vectors
  - R_β aligns normal vectors
- R_α simple rotation to align 2 vectors
  - $v_1 = B - A, v_2 = B' - A'$
  - $\omega_\alpha = v_1 \times v_2$
  - $\alpha = \cos^{-1}(v_1 \cdot v_2)$
- R_β found by rotating angle β around axis v_2
  - $n_P^* = R_\alpha$
  - $n_P, P = A, B$
- $\beta' = \pi - \beta$
- Translation estimated from R

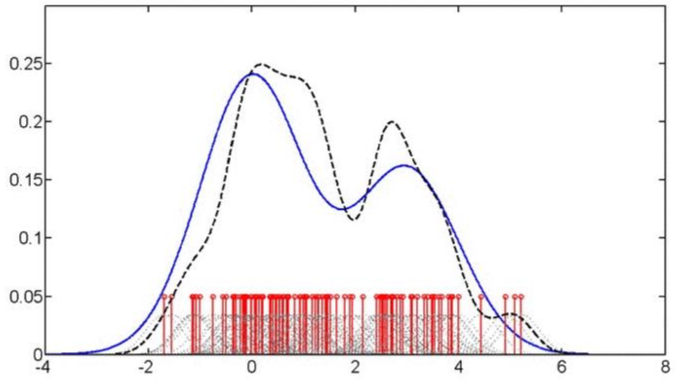


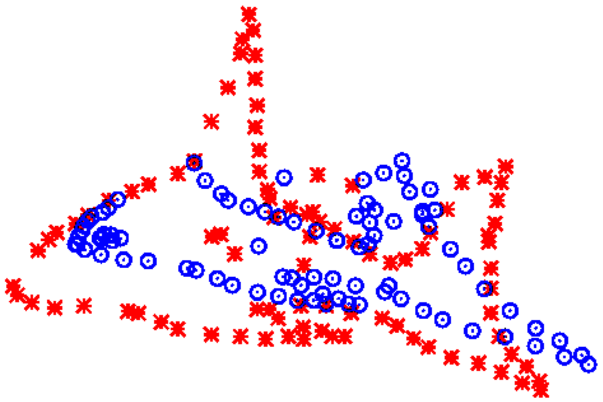


Coherent Point Drift (CPD)



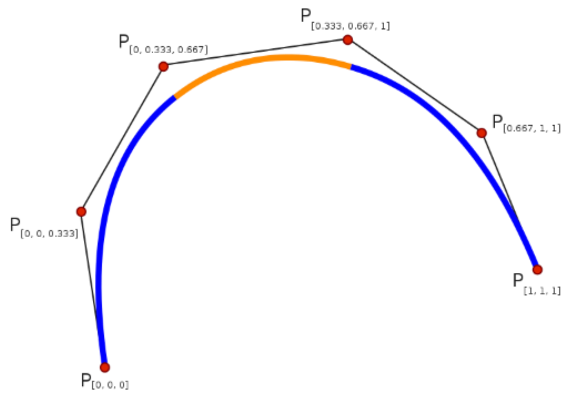
- Probabilistic approach to align point sets.
- Consider problem as probability density estimation problem, fit Gaussian Mixture Model centroids by maximizing likelihood.
- Let $X_{\{M\times d\}} = (x_1, x_2, \dots, x_M)^T$ be the template PC and $Y_{\{N\times d\}} = (y_1, y_2, \dots, y_N)^T$ the target PC
    - d – PC dimension (3)
    - M and N – Amount of points in X and Y.

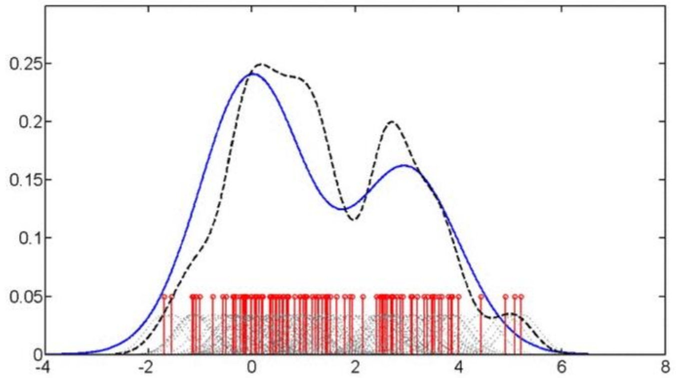


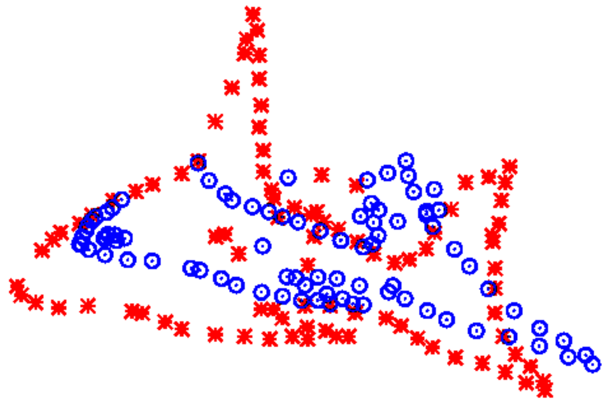


Coherent Point Drift (CPD)

- $X_{\{M\times d\}} = (x_1, x_2, \ldots, x_M)^T$ template PC
- $Y_{\{N\times d\}} = (y_1, y_2, \ldots, y_N)^T$ target PC
- Uses weighted GMM probability density function. Noise or outliers accounted as
  - $p(y) = \omega \frac{1}{N} + (1-\omega)\ \sum_{(m=1)}^{M} \frac{1}{M} p(y|m)$
    - $p(y|m) = \left(\frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}\right) \exp\left(-\left(\frac{||y-x_m||^2}{2\sigma^2}\right)\right)$
  - ω – Weight of uniform distance between 0 to 1.
  - σ – Standard deviation







Coherent Point Drift (CPD)

- Next step – using expectation-maximization scheme (EM) to find final 3D rigid transformation.
- E-step (posterior probability of GMM)
  - $P^{(i)}(m|y_n) = \dfrac{\exp\left(-\frac{1}{2}\left\|\frac{y_n - T\left(x_m, \phi^{(i)}\right)}{\sigma^{(i)}}\right\|^2\right)}{\sum_{k=1}^{M}\exp\left(-\frac{1}{2}\left\|\frac{y_n - T\left(x_m, \phi^{(i)}\right)}{\sigma^{(i)}}\right\|^2\right) + \left(2\pi\left(\sigma^{(i)}\right)^2\right)^{\frac{d}{2}}\frac{\omega}{1-\omega}\frac{M}{N}}$
- Calculate correspondence probability matrix
  - $P = [P(m|y_n)]_{M\times N}$

Coherent Point Drift (CPD)

- M-step, new parameter set calculated by maximizing auxiliary function $Q\left(\Theta, \Theta^{(i)}\right)$, upper bound for log-likelihood function
- $L(\Theta) = \log(\Pi_{n=1}^{N}\left(p(y_n)\right) = \sum_{n=1}^{N} \log \sum_{m=1}^{M+1} P(m)p(y_n|m)$

Accelerated Coherent Point Drift (ACPD)

- CPD suffers from high computational complexity / convergence to local minima.
- ACPD offers 2 methods to speed up performance

Accelerated Coherent Point Drift (ACPD)

- 1. Speed up Expectation-Maximization
  - gSQUAREM
- 2. Calculating probability matrix P more efficiently
  - DT-IFGT
  - Reduces $O(M \cdot N)\ to\ O(M + N)$