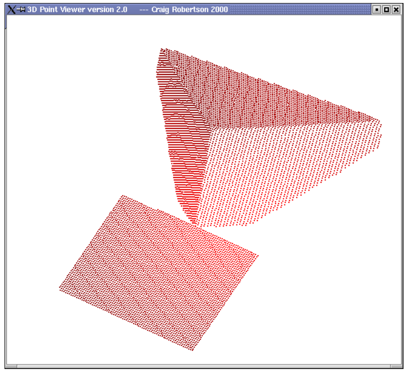
System 3 Introduction

Is there a Wedge in this 3D scene?



Data a set of 3D points!

System 3 Overview

3D part recognition using range data

1. Range data from light stripe triangulation
2. Differential geometry of surfaces
3. Extraction of planes from range data via region growing
4. 3D geometric modeling
5. Model-data matching
6. 3D pose estimation
7. Verification

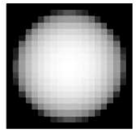
Range Data

Intensity image: observed_brightness(r,c)

Range image: distance_from_sensor(r,c) or $\{(x_i, y_i, z_i)\}$

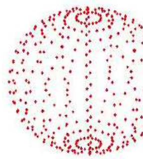
top: intensity bottom: range

Range Data Representations



Range image:

- (r,c) pixel location
- pixel encodes depth, not colour



Point cloud: $\{(x, y, z)\}$

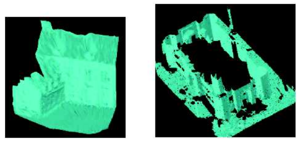
Active 3D Sensing - Motivations

Parts/Objects:

- Analysis/manufacture
- Reverse engineering

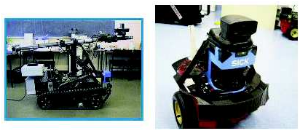
Buildings:

- Use in 3D VR
- Change analysis



Robotic navigation:

on-board laser scanner



Why Range Data

Advantages

Direct, accurate 3D scene information

Unambiguous measurement (unlike brightness)

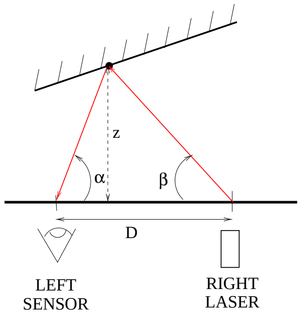
Disadvantages

More complex/expensive sensor

Dark/shiny objects a problem

Generally indirect capture (eg. computed, scanned)

Triangulation range sensors



$z = f(\alpha, \beta, D)$

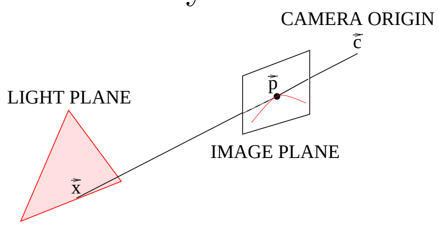
Light beam usually a laser (“laser range scanning”):

Bright
Single frequency (eg 633 nm)
Matching optical filter can eliminate other scene light

Triangulation range calculation

Find pixel $\vec{p}$ on laser stripe (here $\vec{p}$ is in 3D coordinates, known from camera parameters)

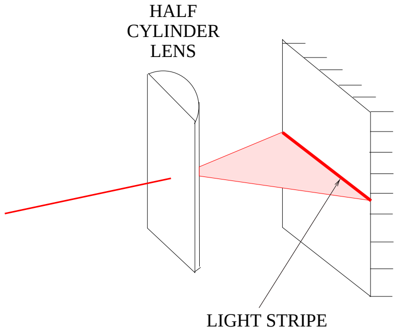
$\vec{p}$ defines ray thru camera origin $\vec{c}$



Ray equation: $\vec{x} = \vec{c} + \lambda(\vec{p} - \vec{c})$
Light plane equation: $\vec{x} \cdot \vec{n} = d$
Find intersection, solve for λ,
substitute to get $\vec{x}$ (3D coords of point)

Getting a full range image

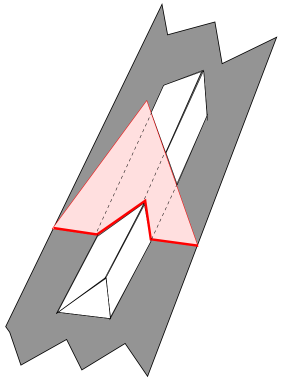
Laser gives a spot, not full image
Use half-cylindrical lens



This gives a stripe on the observed target
For full range image, need to cover all of target

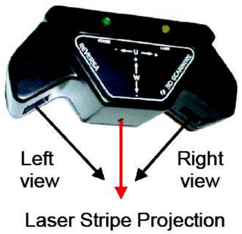
Covering the whole scene

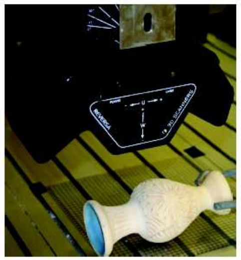
1) Can sweep light plane with rotating mirror
2) Can move sensor (eg sensor in lab)
3) Move parts underneath stripe, eg on a conveyor belt



Builds up image column by column as part moves

Example: Reversa 25 Range Scanner

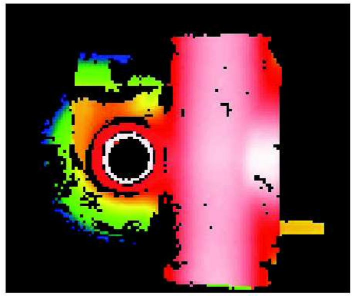




Laser scan head mounted on XYZ robotic gantry

- Accuracy X/Y: 0.05mm, Z(depth): 10 μm
- Cost c. £50,000
- Flat bed object capture via dual camera triangulation

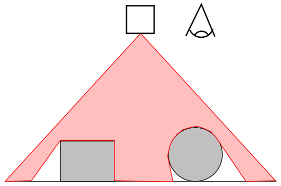
Example Scans



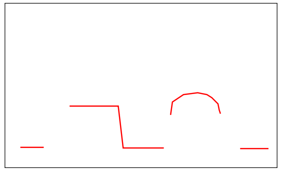
Point cloud (left) and depth coded range image (right)

Problem of Observed stripe

If scene scanned from above:



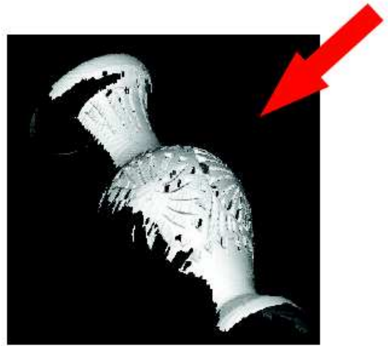
The TV camera sees:



Each row r corresponds to a different depth z(r)
Gives a linear set of range values

Incomplete data

Have depth/3D knowledge in only 1 direction:

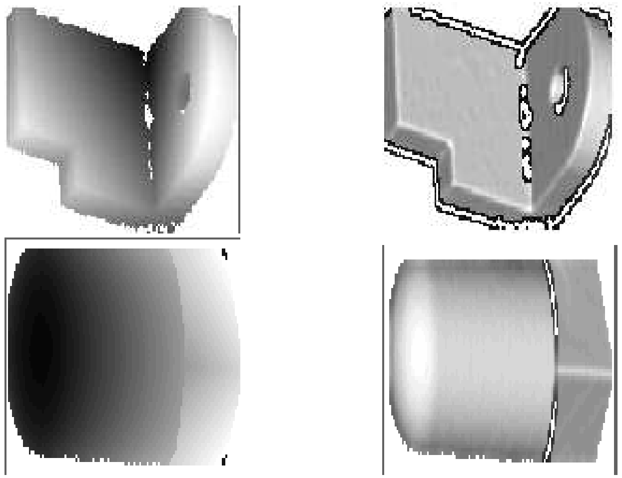


Possible solutions (both difficult):

- Capture from different directions and merge
- Infer missing data from observed data

Range image examples

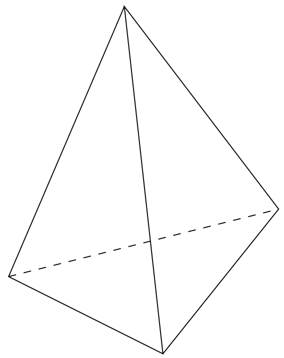
Raw range image Cosine shaded



More range image examples

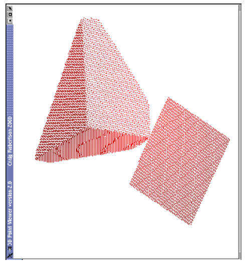
Midlecture Problem

What would a range image of this object look like if the sensor was above this part?



Segmentation: Plane Surface Extraction

Assume: scene contains only planes



Aim: extract instances of planes

- Can be used for later part recognition
- Local shape classes are too noisy
- Use surface fitting instead of diff. geom.

Planar Segmentation Algorithm

Range image *versus* point clouds

Row×Column image representation

- Obvious neighbour relations
- Easier region growing algorithms

3D Point Clouds

- Neighbour relations in R^3
- Good data structures can help with neighbour connections

Segmenting range image into planar regions:
Use region growing algorithm

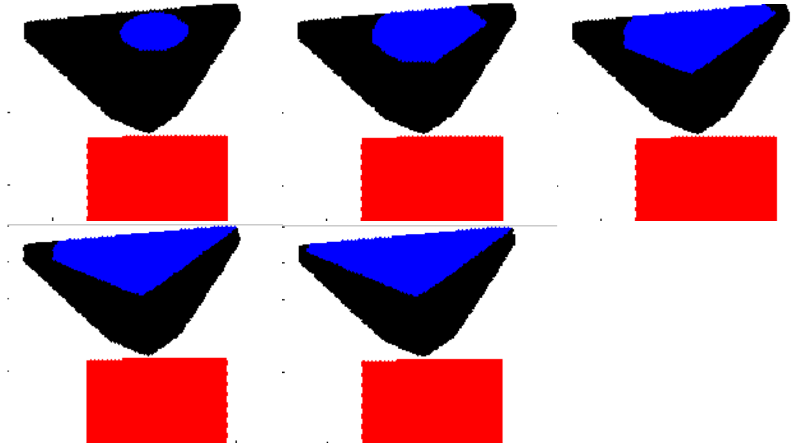
Surface Detection Main Algorithm

```
% find surface patches
[NPts,W] = size(R);
planelist = zeros(20,4);
foundcount=0;
while notdone

  % select small local surface patch from remaining points
  [oldlist,plane] = select_patch(remaining);

  % grow patch
  stillgrowing = 1;
  while stillgrowing

    % find neighbouring points that lie in plane
    stillgrowing = 0;
```

```
[newlist,remaining] = getallpoints(plane,oldlist,
                                   remaining,NPts);
[NewL,W] = size(newlist);
[OldL,W] = size(oldlist);
if NewL > OldL + 50
  % refit plane
  [newplane,fit] = fitplane(newlist);
  if fit > 0.04*NewL % fit going bad - stop growing
    break
  end
  stillgrowing = 1;
  foundcount = foundcount+1;
  planelist(foundcount,:) = newplane';
  oldlist = newlist;
  plane = newplane;
```


Region Growing Principles

Given a planar region formed from points S with equation $\vec{n}'\vec{x} + d = 0$, and a new point $\vec{y}$, add $\vec{y}$ to S if:
1) $| \vec{n}'\vec{y}+d | < \tau_p$ and 2) there is a point $\vec{z}$ in S such that $|| \vec{y}-\vec{z} || < \tau_n$.



Plane Fitting

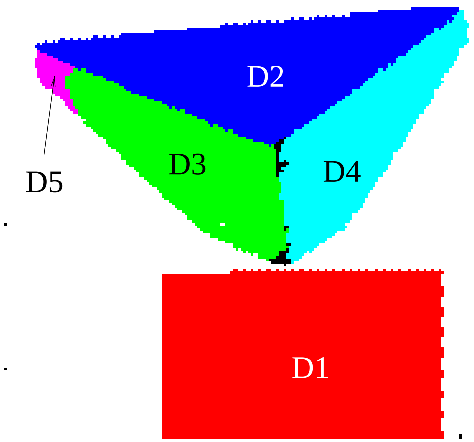
Given a set of datapoints $\{\vec{x}_i\}$, find the $\vec{n}$ and d that best fit $\vec{n}'\vec{x}_i + d = 0$ for all i.

Extend data: $\vec{y}_i = [\vec{x}_i, 1]$
Extend parameters: $\vec{p} = [\vec{n}, d]$
Plane equation is now: $\vec{y}_i'\vec{p} = 0$

Least squared error:
$\sum_i(\vec{y}_i'\vec{p})^2 = \sum_i \vec{p}'\vec{y}_i\vec{y}_i'\vec{p} = \vec{p}'(\sum_i \vec{y}_i\vec{y}_i')\vec{p} = \vec{p}'M\vec{p}$

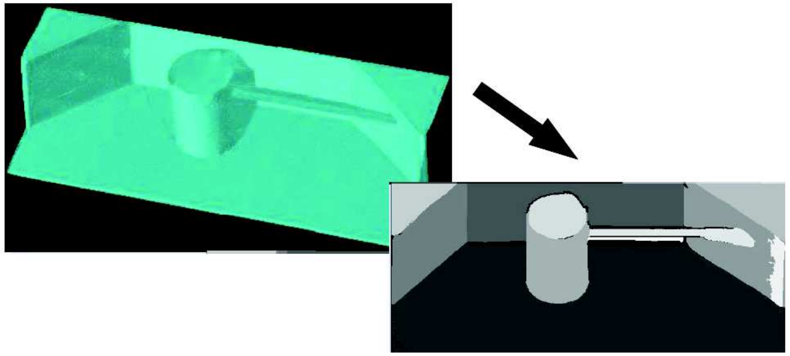
Eigenvector of smallest eigenvalue of M is desired parameter vector, provided eigenvalue is small.

Full Segmentation



Could get 4 planes by parameter adjustment, but 5 means more data for matching stage

Extensions



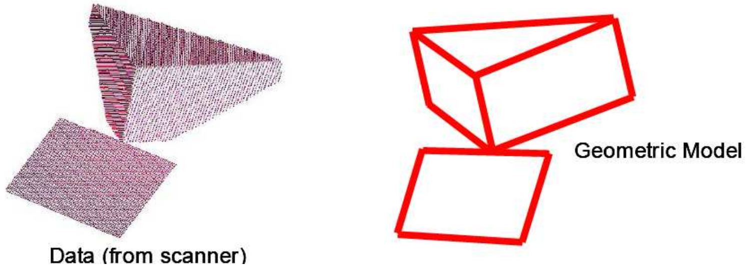
Extend fitting to additional surface types: cylinders, spheres, etc

Allows recognition of more complex objects

School of Informatics, University of Edinburgh

3D Geometric Modelling

Goal: model 3D objects for recognition



Recognition requires some sort of model

Easier matching if data and model use same representations

School of Informatics, University of Edinburgh

3D Coordinate Systems

Like 2D systems: for modelling and object pose

Need rotation and translation specification
Translation easy - 3D vector $\vec{t} = (t_x, t_y, t_z)'$

Rotation needs 3 values. Many different coding systems.

Altogether, 6 degrees of freedom = 6 position parameters.

School of Informatics, University of Edinburgh

Typical Rotation Specification

Arbitrary angle order, so specify rotation as:

$$\mathrm{R} = \mathrm{R}_x(\theta_x)\mathrm{R}_y(\theta_y)\mathrm{R}_z(\theta_z)$$

Where

$$\mathrm{R}_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) \\ 0 & sin(\theta_x) & cos(\theta_x) \end{bmatrix}$$

$$\mathrm{R}_y(\theta_y) = \begin{bmatrix} cos(\theta_y) & 0 & -sin(\theta_y) \\ 0 & 1 & 0 \\ sin(\theta_y) & 0 & cos(\theta_y) \end{bmatrix}$$

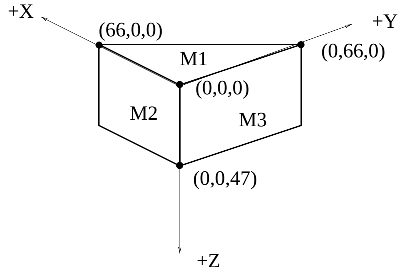
$$R_z(\theta_z) = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation parameters are: $\{\theta_x, \theta_y, \theta_z\}$

Other systems possible: yaw/pitch/roll, azimuth/elevation/twist
Different parameter values, but always the same rotation, when encoded in matrix R
Object position/translation: vector in R^3

3D Shape Modelling

Similar to 2D Modelling
Needs 3D coordinate system + 3D shape primitives



Our primitives: polyhedra, defined by polygonal patches, defined by lists of edges

Wireframe modelling

Representation Scheme

Model: set of polygons (object faces)

Polygons: set of edges (polyhedron edges)

Edge: 2 points in R^3 (edge endpoints)

Wedge Model

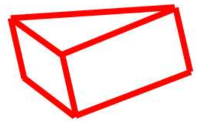
```
planenorm(1,:) = [0,0,-1];       % tri face 1 surf normal
facelines(1) = 3;                % # of boundary lines
model(1,1,:) = [0,0,0,66,0,0];   % Edge 1
model(1,2,:) = [0,0,0,0,66,0];   % edge 2
model(1,3,:) = [0,66,0,66,0,0];  % edge 3
planenorm(2,:) = [0, -1, 0];     % rect face 2 surf normal
facelines(2) = 4;
model(2,1,:) = [0,0,0,0,0,47];
model(2,2,:) = [0,0,0,66,0,0];
model(2,3,:) = [66,0,0,66,0,47];
model(2,4,:) = [0,0,47,66,0,47];
planenorm(3,:) = [-1, 0, 0];     % rect face 3 surf normal
facelines(3) = 4;
model(3,1,:) = [0,0,0,0,0,47];
...
```

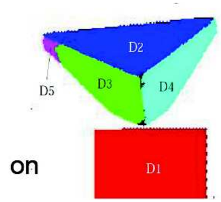
Midlecture Problem

How would you model the visible portion of a cube?

3D Recognition

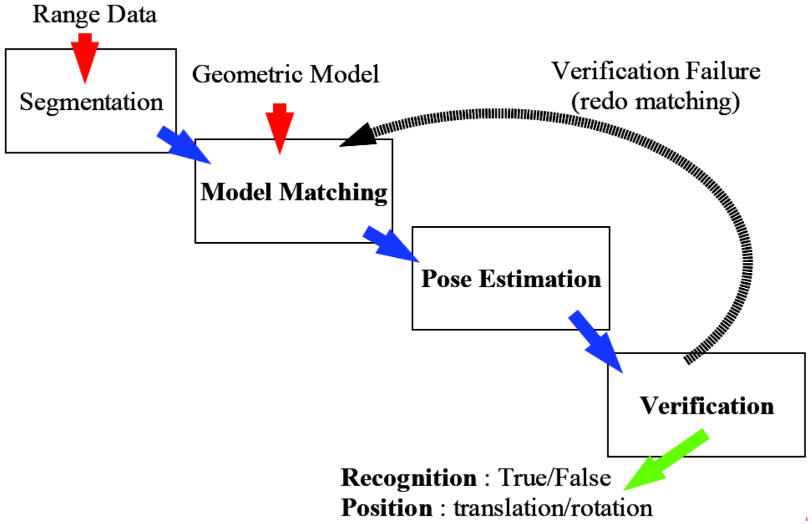
Is there a wedge in the scene?





- Have geometric model: 3D *a priori* knowledge
- Data from laser scanner
- Planar region segments
- Geometric transformations

Range data: 3D Recognition Pipeline



Recognition: Model Matching

Use Interpretation Tree

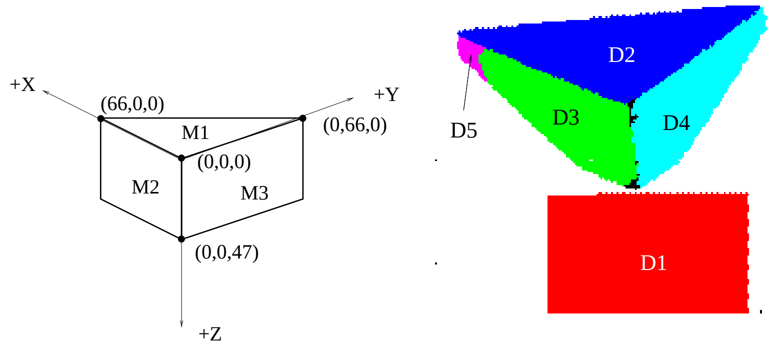
Unary constraint: eg. surface area

Binary constraint: eg. angle between vectors, like surface normals

Trinary constraint: sign of vector triple product $\vec{a} \cdot (\vec{b} \times \vec{c})$, eg. on surface normals

Result: paired model and data planes

Recognition: Matching Results




```
matchedpairs =
   model   data
    M1      D2
    M2      D3
    M3      D4
```


Pose Estimation

Like 2D case, estimate rotation first, then translation

Assume:

- N paired planes $\{(M_i, D_i)\}_{i=1}^{N}$
- model and data normals $\{\vec{m}_i\}$ and $\{\vec{d}_i\}$
- a point on each model patch $\{\vec{a}_i\}$
- a point on each data patch $\{\vec{b}_i\}$ (need not correspond to $\vec{a}_i$)

Rotation Estimation

Want R such that $\mathrm{R}\vec{m}_i \doteq \vec{d}_i$

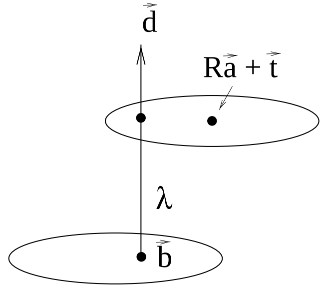
A least square problem, minimizing

$$\sum_i || \mathrm{R}\vec{m}_i - \vec{d}_i ||^2$$

Form matrix $\mathrm{M} = [\vec{m}_1\vec{m}_2 \ldots \vec{m}_N]$
Form matrix $\mathrm{D} = [\vec{d}_1\vec{d}_2 \ldots \vec{d}_N]$
Compute singular value decomposition (SVD):
  svd(DM') = U*S*V'
Compute rotation matrix: R = V*U'
  Assumes at least 3 non-coplaner vectors
  (caution 1 special case)

Translation Estimation

Minimize the perpendicular separation λ_i between rotated model patch and data patch:



Goal: find $\vec{t}$ that minimizes $\sum_i \lambda_i^2$
Form matrix: $\mathrm{L} = \sum_i \vec{d}_i\vec{d}_i'$
Form vector: $\vec{n} = \sum_i \vec{d}_i\vec{d}_i'(\mathrm{R}\vec{a}_i - \vec{b}_i)$
Compute translation $\vec{t} = -(\mathrm{L})^{-1}\vec{n}$

Verification

Multiple possible matching solutions

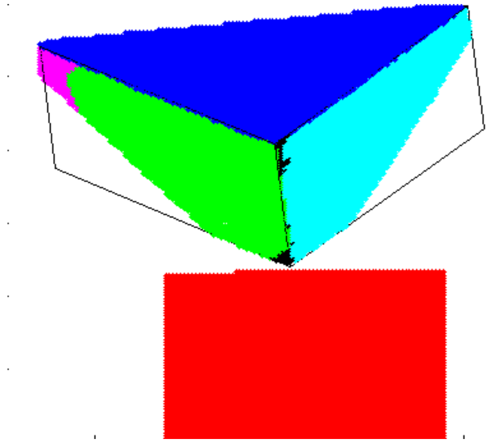
globally invalid pairings, alternative pose hypotheses

Use verification to find correct one

1. Rotated model normals $\vec{m}_i$ close to data normals $\vec{d}_i$: $acos(\vec{d}_i' \mathrm{R} \vec{m}_i) < \tau_1$
2. Transformed model vertices $\vec{e}_i$ lie on the data plane $\vec{n}'\vec{x} + d = 0$: $\mid \vec{n}'\vec{e}_i + d \mid < \tau_2$

Matching Results

Object recognized but three pose solutions as verification didn't check overlap areas



Accurate model-data alignment!

Range data: edges

Edges originate in range data from:

- Changes in depth: **blade edge**
- Changes in surface orientation: **fold edge**
- Changes in surface curvature properties

Blade and fold edges also usable for recognition
Similar to 2D case. See more later with stereo

Discussion

- Range sensors now commercially available: we designed a £50 sensor, commercial starts at a few 1000 pounds.
- Accuracy can be amazing: our commercial sensor has 10 μm accuracy.
- Range data unambiguous and very useful: gives 3D info directly rather than needing inference from other data

- Many different ways to segment data patches, many sensitive to data noise and slow.
- Much more efficient to segment if data is in image array rather than a set of points
- Techniques presented here particularly useful in an industrial or robot navigation context

What We Have Learned

- Range image and 3D point cloud data
- Triangulation range sensor technology
- Least square planar surface fitting
- Region growing
- 3D coordinate systems and transformation specification
- 3D wire frame shape modelling
- 3D pose estimation