CITS 4402 Computer Vision

Prof Ajmal Mian

Lecture 12 – 3D Shape Analysis & Matching
Overview of this lecture

- Revision of 3D shape acquisition techniques
- Representation of 3D data
- Applying 2D image techniques on 3D data
- ICP Algorithm
- Keypoint detection in 3D data
- 3D feature extraction and matching
Laser stripe scanner

- A laser stripe is moved over the object
- The stripe is orthogonal to the epipolar lines and helps resolve correspondence
- Can use one camera and one laser stripe projector
Minolta Vivid 3D scanner uses laser stripe

- Available in CSSE

- Fine scan resolution is 640x480 with 1mm resolution

- Fast scan is 320x240 but scans in less than a second
Structured light

- Multiple stripes or other 2D patterns are simultaneously projected

- Stripes are coded through
  - Time coding
  - Colour coding

- Other spatial patterns
  - Random pattern

- Kinect 1 projects spatial infra-red random pattern
Sample 3D data from Kinect 1

Comparing with other scanners

<table>
<thead>
<tr>
<th>Bosphorus (InSpeck)</th>
<th>FRGC (Minolta)</th>
<th>CurtinFaces (Kinect)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Bosphorus" /></td>
<td><img src="image" alt="FRGC" /></td>
<td><img src="image" alt="CurtinFaces" /></td>
</tr>
</tbody>
</table>

![Images of 3D data from different scanners]
Full 3D reconstruction with Kinect 1

- Rotate object
- Get 30 fps
- Integrate
- Kinect SDK is available
- OpenCV + Matlab
Laser time of flight scanner

- Fire laser and measure its time delay (phase shift) after reflection from an object
- Measure distance in that directions
- Kinect 2 is an example
- Has better spatial resolution (number of points)
- Has better depth resolution than Kinect 1
Representing 3D data

- Raw 3D data is basically a collection of points called pointcloud.
- Pointcloud is an $3 \times N$ matrix of the $[X, Y, Z]^T$ coordinates of the $N$ scene points that were scanned.
- A pointcloud can be visualized using rendering softwares based on OpenGL.
- Pointclouds are converted to mesh by connecting nearest points to form triangles (or polygons).
- Polygons are filled and shaded to render surfaces.
Pointcloud of UWA Winthrop Hall

Scanned with laser time of flight scanner
Basic representation of 3D data

Pointcloud

Mesh

Surface rendering
Pointcloud example

Matlab: plot3.m

```matlab
mesh = readPLYfile('c:\chef.ply');
X = mesh.vertices(:,1);
Y = mesh.vertices(:,2);
Z = mesh.vertices(:,3);
plot3(X,Y,Z,'.');
axis equal;
rotate3d;
```

www.csse.uwa.edu.au/~ajmal/code.html
Mesh example

```matlab
mesh = readPLYfile('c:\chef.ply');
X = mesh.vertices(:,1);
Y = mesh.vertices(:,2);
Z = mesh.vertices(:,3);
trimesh(mesh.triangles,X,Y,Z);
axis equal;
rotate3d;
```
Surface

```
mesh = readPLYfile('c:\chef.ply');
X = mesh.vertices(:,1);
Y = mesh.vertices(:,2);
Z = mesh.vertices(:,3);
trisurf(mesh.triangles,X,Y,Z);
axis equal;
rotate3d;
```
Rendering as a shaded surface

\[ M = \text{calcMeshNormals}(\text{mesh}); \]
\[ N = M.\text{vertexNormals}; \]
\[ [~, p, ~] = \text{cart2sph}(N(:, 1), N(:, 2), N(:, 3)); \]
\[ \text{trisurf(mesh.triangles, } X, Y, Z, p); \]
\[ \text{axis equal; } \]
\[ \text{shading interp; } \]
\[ \text{colormap gray; } \]
\[ \text{rotate3d;} \]

www.csse.uwa.edu.au/~ajmal/code.html
How to make full 3D models?

Problem: A scanner can only see part of the object’s surface
- A single view scan is also referred to as 2.5D
- Strictly speaking, a single Kinect frame is 2.5D

Solution: Scan the object multiple times from different angles and integrate

Before we can integrate scans, we need to register/align them
Iterative Closest Point (ICP) algorithm

Given two pointclouds $P_1$ and $P_2$, we want to rotate and translate $P_2$ so that it aligns with $P_1$.

For each point in $P_2$, find its nearest point in $P_1$ (Correspondences).

Remove incompatible points based on
- Distance, colour, normals, curvatures, boundary points.

Let $Y \subset P_2$ and $X \subset P_1$ be the remaining set of corresponding points.

Minimize $\|RY + t - X\|_2$ where $R$ is the rotation matrix and $t$ is the translation vector.
ICP algorithm

- Find correspondences $X$ and $Y$
- Subtract mean: $\bar{Y} = Y - m$, and $\bar{X} = X - n$
- Perform SVD: $USV^T = \bar{Y} \bar{X}^T$
- $R = VU^T$ (Ensure that $\det(R) = 1$)
- $t = n - Rm$
- Repeat until $\|RY + t - X\|_2 < \epsilon$
- Need minimum of 3 points to solve for $R$

www.csse.uwa.edu.au/~ajmal/code.html
ICP algorithm limitations

- The two pointclouds should be close, otherwise ICP will fail

- If they are not close, the first set of correspondences must be provided
  - Manually
  - Or by automatic feature matching

- Can still fail when
  - The surface is rotationally symmetric
  - The surface has repetitive structure
ICP registration example

Three corresponding points marked to find initial coarse registration

After ICP registration
Full 3D model construction

- Register multiple scans

- Integrate them to form a single smooth surface (outside the scope of this lecture)
ICP applications

1. Surface alignment/registration

2. Surface matching

- The final value of error $\epsilon$ is used to decide how well the two surfaces match

- Advantages
  - Partial surface matching
  - Inbuilt correction of alignment

- Drawbacks
  - Computationally expensive
Mesh representation

- Many scanners output pointclouds as 4 matrices of the same size
  - $X$ matrix containing the x-coordinates at each ‘pixel’ location
  - $Y$ matrix containing the y-coordinates at each ‘pixel’ location
  - $Z$ matrix containing the z-coordinates at each ‘pixel’ location
  - $M$ matrix of binary values with 1 at valid and 0 at invalid pixel

- Kinect 2 gives XYZ RGB pointcloud

- Since the pointcloud is arranged on a planar grid, a simple 2D delaunay triangulation of the XY coordinates will give you the mesh

- Delaunay triangulation connects nearest three points
  - validPixels = find(M(:));
  - triangles = delaunay(X(validPixels), Y(validPixels));
  - Vertices = [X(validPixels) Y(validPixels) Z(validPixels)];

- Remember to remove elongated triangles
Example: Pointcloud to mesh conversion

- The `triangles` variable will contain sets of 3 points (their index numbers) that make a triangle.
2D-grid based representations of pointclouds

- Depth image:
  - Depth of every point from a plane
  - The $Z$ matrix can be regarded as the depth image
  - For actual depth image, need to resample on a uniform XY grid
  - We can display/visualize it using image visualization (imshow, imagesc)
  - Need to scale $Z$ values so that they are 0-255

- Range image:
  - Range of every point from the camera center

- The difference is small and they are used interchangeably in the literature

- Kinect 2 can also give depth images
Example depth images

- Bright pixels mean close points
Principal curvatures

- Surface normal at a point

- Maximum curvature $\kappa_1$

- Minimum curvature $\kappa_2$

- Principal directions

- Gaussian curvature $\kappa_1 \kappa_2$

- Average curvature $(\kappa_1 + \kappa_2)/2$
More 2D-grid based representations of pointclouds

- We can derive many other 2D-grid based representations from the $X, Y, Z$ matrices

- The normal vector $[n_x, n_y, n_z]^T$ gives 3 images. The normal vector $[\theta, \phi, r]^T$ gives another 2 images. Minimum and maximum curvatures give another 2 images.
Applying 2D Gaussian on the Z-image

- 7x7 window and $\sigma = 4$

Elongated triangles after delaunay triangulation
Need to remove spikes and fill holes before smoothing

- Smoothing with a median filter
Gradient images from Z-image

Range image  X gradient  Y gradient
Detecting features in 3D data

- Train Haar classifier on gradient images
- Detect features in real-time
Spherical feature

- Sum the number of points inside spheres of increasing radii

- Easily implemented by histogramming only the radii values in spherical coordinates

- 1-D feature
Spin images (Johnson & Hebert, PAMI’99)

Two steps

1. Use a point and its normal to define an image plane

2. Spin the image plane and sum the points passing through each bin/pixel

Easily implemented by histogramming in cylindrical coordinates. Download code from

www.csse.uwa.edu.au/~ajmal/code.html
Tensor feature

- Define a 3D grid over the mesh
- Need three non-coplanar points OR two points and their normals
- Find the area of intersection with each cubelet
- 3D feature

Mian et al, “3D object recognition and segmentation in cluttered scenes, PAMI 2005
Feature comparison

- Spherical feature does not need any reference vector or coordinate basis
  - Most robust
  - Less descriptive

- Spin image needs a reference normal vector
  - Medium robust
  - Medium descriptive

- Tensor feature needs a complete 3D reference frame
  - Least robust
  - Most descriptive
Shape index

\[ SI = \frac{2}{\pi} \arctan \frac{\kappa_2 + \kappa_1}{\kappa_2 + \kappa_1} \]

Shape Index image

- Can organize SI values into an image
- And use standard image matching

P Szeptycki et al., "A coarse-to-fine curvature analysis-based rotation invariant 3D face landmarking" BTAS 2009.
Keypoint detection in 3D images

- Similar to corner detection and LoG/DoG blob detection
- Need to detect interesting points
- To avoid feature extraction at every point
  - Save computation
  - All locations are not discriminative

Does this give you an idea to find keypoints?
Keypoint detection in 3D images

- Can apply image based techniques on Z-image

- Problems
  - Does not fully exploit the potential of 3D data
  - 3D surfaces are smooth (corners and edges are subtle)
  - Corners and edges are at boundary or at noisy locations (e.g. spikes)

- With rotation, the pixel values
  - change in depth images
  - move in conventional 2D images
Keypoint Quality (KPQ) : Feature detection in 3D pointclouds

- Given a point $p$, crop out a spherical region with radius $r$ around it
- Let $P$ be the $3 \times N$ matrix of $N$ points inside the sphere
- Find its covariance matrix $C = (P - m)(P - m)^T$
- Perform PCA: $CV = DV$
- Align the points on their principal axes: $P' = V(P - m)$
- Use the X, Y coordinates of the aligned points to find the ratio $\delta = \frac{\max(X) - \min(X)}{\max(Y) - \min(Y)}$
- If $\delta > \text{threshold}$, accept the point as a keypoint
- OR $\frac{\lambda_1}{\lambda_2} > \text{threshold}$
KPQ Intuition

- KPQ selects keypoints where the surface curvature is different in the two directions

- Unlike edge detection or Lucas Kanade where $\lambda_1 \approx \lambda_2$ is preferred, in KPQ such points are not preferred as they lie on spherical or flat surfaces

- A unique coordinate basis cannot be defined at a point where $\lambda_1 \approx \lambda_2$ because the principal directions are not unique or sensitive to noise
KPQ: Number of keypoints vs threshold
KPQ feature extraction

- A smooth surface is fitted to the points in $P'$
- Depth values w.r.t. the center are vectorized and used as a feature
- We can align the corresponding texture to extract invariant features
KPQ: Automatic scale selection

- What should be the radius \( r \) i.e. the scale for keypoint detection and feature extraction?

- Perform scale space search

- Use different values and select the scale/radius \( r \) at which \( \lambda_1/\lambda_2 \) is max

- Discard point where a max cannot be found
KPQ face recognition

- Similar faces will have more and spatially coherent matches

- Different faces will have fewer and spatially incoherent matches
KPQ scale invariant object recognition

- Match KPQ features to find objects in cluttered scenes at unknown scales
- Prune geometrically inconsistent matches
Scene completion and match verification

- Align the database model with the scene

- Verify the match
  - ICP error
  - Free space violation
  - Occupied space violation

- ICP error should be small

- The aligned model should not block visible surfaces

- The aligned models should not cross over into other objects
Summarizing the 3D shape matching pipeline

1. Detect keypoints and their scale
2. Extract features at the keypoints
   (KPQ feature, spherical, spin image, tensor, SI, curvatures ……………)
3. Match features
   ($\ell_2$ norm, dot product, ….)
4. Prune matches
   (remove geometrically impossible matches)
5. Verify matches
   (ICP, free/occupied space violation, ….)
6. Output best match
Can we recognize actions?

- We know how to match objects in images
- We know how to match objects in 3D data
- What about video?
- Can we recognize an action rather than the object/person performing that action?
- Space-time matching or spatiotemporal matching
- Space-time SIFT has been proposed but we will focus on 3D data
3D spatiotemporal matching

- Sometimes referred to as 4D in the literature

**Motivations**

- 3D data is preferred for action recognition because it can capture all 6 degrees of freedom (3 rotations and 3 translations)
- 3D avoids texture which can cause errors in action recognition
- Kinect 1 and Kinect 2 are available at a very low cost
Human action recognition from Kinect skeleton data

- Kinect gives depth image and skeleton data at 30 frames/second
- Can be used for action recognition
Histogram of Depth Gradients

- Recall Histogram of Oriented Gradients (HOG)
- HOG features can be extracted from gradient of depth images

HOG $Z_x \ Z_y \ Z_t$

Skeleton displacement feature
Generating novels views from 3D models

- Can generate novel views by rotating the object and re-rendering it
- Can generate novel illuminations
Demos

- See recording of lecture 13 for the following demos

- SIFT matching

- Single image based height measurement

- Microsoft Kinect 2 sensor
  - 3D Shape capture
  - Body skeleton tracking
  - Infra-red capture
Summary

- Revision of 3D shape acquisition techniques
- Representation of 3D data
- Applying 2D image techniques on 3D data
- ICP Algorithm
- Keypoint detection in 3D data
- 3D feature extraction and matching