2D and 3D Vision for Robotics

Vinod Kumar Kurmi
Lakshmi Shraya Koppulla
Nandani Kumari Roma
Siddharth J. Singh
Dr. K. S. Venkatesh

Computer Vision Laboratory
Department of Electrical Engineering
Indian Institute of Technology Kanpur
Outline

• Introduction
• Presentation
  • Hand gesture recognition
  • High speed shape estimation
• Demos
  • 3D sensor capture and reconstruction
  • Display of 3D information with perspective effects
• Exercises
  • Color based object detection
  • Shape based object detection
  • Shape and color object detection
A gesture is generally defined as any motion or configuration of human body parts, that carries some information. For example, waving the hand carries the information goodbye.

In computer vision, the hand gesture recognition can be treated as a combination of three sub parts hand detection, tracking the hand and gesture recognition.

The two most popular vision based approaches in the hand gesture recognition are marker based analysis and marker less approaches.
Every gesture recognition system is characterized by its unique features.

The essential task is to select these features from the vision based data.

For detecting the features of hand, there are two techniques available.

- Model Based
- Appearance Based
Proposed Gesture Recognition System

We present a novel approach in which no hand model or trained data set is required.

Hand segmentation is carried out with the assistance of a depth histogram and this segmentation is independent of clutter of any kind and lighting conditions.

Isolation of the hand from other objects in the scene is achieved by using the fingertip constellation, which leads to very low computational activity required for hand segmentation.

The system easily works in real time.

Gestures are detected by the change in hand depth or orientation or position with reference to an initial state.

Our gesture vocabulary is intuitive and comfortable. It doesn’t matter if the gestures are performed in seated or standing pose. Invalid gestures are automatically rejected and valid ones are acted upon.

The gestures are scalable in number.
The depth map is an image that contains the distance information from the sensor. Each pixel value is mapped as a function of distance from the depth sensor.
In our dissertation, we have extracted depth data using Kinect sensor. Kinect sensor projects a pseudo-random pattern on the objects using IR rays, to calculate their depth.

The depth data sent by the Kinect is of 16 bits, with the least significant 3 bits containing index of user detected at pixel and 1 bit for error, the remaining 12 bits carry the depth information.

\[
\text{Actual distance (depth) in mm} = \text{pixel value} \times \text{depth scale factor}
\]

where depth scale factor: \(2^{12}/255\)
Depth Histogram

Depth histogram is a graphical representation of the distribution of pixels as a function of depth.

While performing gestures, the hand should always be within camera view range and at a depth value lying in between the camera and the human body.

Figure: Depth Histogram
Depth Thresholding

There is a valley between the hand depth peak and the human body depth peak in this histogram. Suppose that the first peak of depth histogram occurs at depth $d_{peak}$ and valley point at the depth $d_{valley}$, then the threshold $t$ is defined as:

$$t = d_{valley} - d_{peak}$$

Now, depth segmentation is done as follows:

$$d(x, y) = \begin{cases} 1 & \text{if } (d_{peak} - t) < d(x, y) < (d_{peak} + t) \\ 0 & \text{otherwise} \end{cases}$$

(2)
The segmented depth sliced images are converted into binary images. In the binary image, pixel value at any position is 1 if object is present at that depth range, otherwise it contains 0 value.

Figure: Segmented Hand
Contour Detection

Contour is a collection of the boundary points that belong to an object.

A contour also gives the information about the area of an object. We have used an OpenCV algorithm to detect contours of the objects in any image.

Figure: Contours of segmented region
Convex Hull and Convexity Defects

We approximated the hand shape with polygon using the OpenCV library function.

Using these convex hull points, we calculate the convexity defects points with the corresponding depth (valley height), starting and ending points of particular defects.

Figure: Depth image
Fingertip constellation of the hand is the basis for identification of hand.

In the open hand, fingertips are always arranged in their unique configuration in space.

Human hand always lies within a specific range of area (1000-8000 pixels).

Human hand has five tip points, and between two tip points there is a convexity defect point (except between the thumb and the little finger).

The angle between two consecutive tip points with respect to the shared convexity defect depth point is always less than 60° (between the thumb and the index finger it goes to 90°).
Orientation of Hand

Orientation of hand is decided with the assistance of Maximum inscribed circle that can accommodate the palm polygon.

A circle $S$ is called Maximum inscribed circle for a polygon $P$, if it is completely lying inside the polygon and any other circle $S_1 \subset P$ does not contain $S$.

Point $s_c$, the center point of contour always lies towards the finger as compared to point $s_m$, the center of Maximum inscribed circle.

The direction vector $p$ from $s_m$ to $s_c$ gives us the orientation of the hand. If the angle component of direction vector $p$ is around $90^\circ$ then the hand is in vertically up position, and if the angle is around $0^\circ$, the hand is in horizontal position.
Maximum Inscribed Circle

Figure: Maximum inscribed circle
Figure: Vertical Orientation
Left Hand and Right Hand

Figure: Horizontal Orientation (a) Left hand. (b) Right Hand
Fingertip Constellation in RGB Image

The objects in an RGB image appear in the same plane. This prevents us from eliminating objects on the basis of depth. We have tried to filter out the image using the skin color tone criteria, and this results in an image in which there are lesser number of objects as compared to the original one.

Figure : Original RGB image
Now this image is passed to the Canny edge detector, to find the edges point in the scene.

We try to join all the edge points using a distance threshold. The sequence of edge points, which are nearer to each other, define the contour or boundary of the objects.

Figure: Edges using Canny Edges Detector
Now, we again connect the different contour points based on the distance criterion, to make a complete contour of an object. Further, with the help of contour analysis, we can figure out the finger point constellation from the RGB images.

This approach still requires refinement.

Figure: Constellation point of hand
Gestures can be performed using either single hand or both hands.

After segmenting the hand, the next step is to decipher if the gesture is done using one hand or two hands.

If two objects at the same depth satisfy all the properties of hand, then it is a two hand gesture.
Algorithm for detecting the number of hands

Figure: Flowchart for the detection of number of hands
Table: Single hand gestures

<table>
<thead>
<tr>
<th>S.N.</th>
<th>Hand Gesture</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Horizontal right hand moved in upward direction</td>
<td>Channel Up</td>
</tr>
<tr>
<td>2</td>
<td>Horizontal right hand moved in downward direction</td>
<td>Channel Down</td>
</tr>
<tr>
<td>3</td>
<td>Horizontal left hand moved in upward direction</td>
<td>Volume Up</td>
</tr>
<tr>
<td>4</td>
<td>Horizontal left hand moved in downward direction</td>
<td>Volume Down</td>
</tr>
<tr>
<td>5</td>
<td>Vertical hand (left/right) closed</td>
<td>Mute/Unmute</td>
</tr>
<tr>
<td>6</td>
<td>Horizontal hand (left/right) moved towards the sensor</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Single Hand Gestures

**Figure**: Channel Up/Down Gesture

Vind Kumar Kurmi (IIT Kanpur)

Hand Gesture Recognition
Figure: Volume Up/Down Gesture
Single Hand Gestures

Figure: Mute/Unmute Gesture
Algorithm for Single Hand Gestures

Figure: Flow chart for the identification of single hand gestures
Two Hand Gestures

We have defined four static and six dynamic gestures for two hand case. The number of gestures is scalable and many new gestures can be added to the vocabulary.

A gesture is considered dynamic, if at least one of the hands is in motion.

It should however be kept in mind that the two hands do not overlap while performing the gesture. They should be kept at a sufficient distance from each other.
## Two Hand Gestures

Table: Two hand static gestures

<table>
<thead>
<tr>
<th>S.N.</th>
<th>Hand Gesture</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Both the hands are in horizontal orientation and fingers are left side in both the hand</td>
<td>Fast Forward</td>
</tr>
<tr>
<td>2</td>
<td>Both the hands are in horizontal orientation and fingers are right side in both the hand</td>
<td>Fast Rewind</td>
</tr>
<tr>
<td>3</td>
<td>Left hand is vertical but right hand is horizontal</td>
<td>Enter Menu</td>
</tr>
<tr>
<td>4</td>
<td>Right hand is vertical but left hand is horizontal</td>
<td>Exit Menu</td>
</tr>
</tbody>
</table>
Two Hand Gestures

Figure: Fast forward/ Fast rewind

Figure: Enter menu / Exit menu
## Two Hand Gestures

### Table: Two hand dynamic gestures

<table>
<thead>
<tr>
<th>S.N.</th>
<th>Hand Gesture</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Both the hands are moving in an inward direction</td>
<td>Zoom In</td>
</tr>
<tr>
<td>2</td>
<td>Both the hands are moving in an outward direction</td>
<td>Zoom Out</td>
</tr>
<tr>
<td>3</td>
<td>Right hand is static but left hand is moving outward</td>
<td>Next Page</td>
</tr>
<tr>
<td>4</td>
<td>Right hand is static but left hand is moving inward</td>
<td>Previous Page</td>
</tr>
<tr>
<td>5</td>
<td>Left hand is static but right hand is moving outward</td>
<td>Scroll Right</td>
</tr>
<tr>
<td>6</td>
<td>Left hand is static but right hand is moving inward</td>
<td>Scroll left</td>
</tr>
</tbody>
</table>
Two Hand Gestures

Figure: Scroll Left/ Scroll Right
Algorithm for Two hand Gestures

Figure: Flow chart for the identification two hand gestures
Results and Discussion

Hand Segmentation

Figure: Segmentation using the first peak of depth histogram

Figure: Segmentation using the second peak of depth histogram
Spurious Object Present at Same Depth

The objects which are present at the same depth level of the hand, do not get eliminated after depth slicing.

The convex hull points and convexity defects points configuration for them cannot be the same as of the hand. This allows us to eliminate such objects.

Based on the fingertip constellation, human hand is identified and remaining objects are ignored.
Spurious Object Present at foreground.

It is also possible that the first peak of histogram doesn’t satisfy the properties of a human hand. This means the hand is not the nearest object from the depth sensor.

In this case, we have to look at the next depth slice. This process ideally can be continued until the human hand is found, but the range of the camera is limited. In this scenario, the depth histogram peak corresponding to the hand will be very small and hand shape analysis will not be accurate.
Figure: Identification of the hand in the presence of other objects
The performance of gestures not only depends upon the technical or programming details, it also depends upon the way the gesture is performed.

A gesture performed in a wrong way, will result in lesser accuracy as compared to our sample set results.

All the gestures work fine in real time scenarios.

Performance of gestures is independent of lighting conditions.

Gestures can also be performed in any body posture, as against the popularly used NITE Middleware.
Performance Analysis of Confusion between Left and Right Hand

Table: Confusion probability between left and right hand (when hand is vertical)

<table>
<thead>
<tr>
<th></th>
<th>Left Hand</th>
<th>Right Hand</th>
<th>No Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left Hand</td>
<td>19/20</td>
<td>1/20</td>
<td>0/20</td>
</tr>
<tr>
<td>Right Hand</td>
<td>1/20</td>
<td>19/20</td>
<td>0/20</td>
</tr>
</tbody>
</table>

Table: Confusion between left and right hand (when hand is horizontal)

<table>
<thead>
<tr>
<th></th>
<th>Left Hand</th>
<th>Right Hand</th>
<th>No Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left Hand</td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td>Right Hand</td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
</tbody>
</table>
When the hand is vertical, the wrong detection probability is more than when the hand is horizontal. This occurs due to the error in detection of the tip point of the thumb.

This problem can be overcome by maintaining a sufficient distance between the two hands during device initialization.

In the horizontal hand case, this identification is based on the remaining four fingertips, thus decreasing the probability of error.
## Performance Analysis of Single Hand Gestures

### Table: Error in single hand gestures

<table>
<thead>
<tr>
<th></th>
<th>Correct</th>
<th>False</th>
<th>No Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel Up</td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td>Channel Down</td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td>Volume Up</td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td>Volume Down</td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td>Mute/Unmute</td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td>OFF</td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
</tbody>
</table>
Performance Analysis of Two hand Static Gestures

Table: Error in static gestures

<table>
<thead>
<tr>
<th></th>
<th>Correct</th>
<th>False</th>
<th>No Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter Menu</td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td>Exit Menu</td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td>Fast Forward</td>
<td>19/20</td>
<td>0/20</td>
<td>1/20</td>
</tr>
<tr>
<td>Fast Rewind</td>
<td>19/20</td>
<td>0/20</td>
<td>1/20</td>
</tr>
</tbody>
</table>
## Table: Error in dynamic Gestures

<table>
<thead>
<tr>
<th></th>
<th>Correct</th>
<th>False</th>
<th>No Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Zoom In</strong></td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td><strong>Zoom Out</strong></td>
<td>19/20</td>
<td>1/20</td>
<td>0/20</td>
</tr>
<tr>
<td><strong>Next Page</strong></td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td><strong>Previous Page</strong></td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td><strong>Scroll Left</strong></td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td><strong>Scroll Right</strong></td>
<td>20/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
</tbody>
</table>
The whole algorithm works fine if the segmentation of the hand is done properly, and segmentation itself is based on the fingertip constellation. Therefore, while performing the gesture, the fingers must be open in such a way that all its five tips are separated to be recognized by the sensor. This system will not work when the hand is closed.

Since each sensor has its own limitations, beyond a limit (maximum and minimum range), system cannot detect the gestures. For our case, we are using Microsoft Kinect-Xbox-360, and the system working range is from 60 cm to 2.5 metre.
Advantages

No hand model required.

No data learning required.

Works in real time.

Completely independent of lighting conditions.

Completely independent of background clutter.

Works practically error free.

Not affected by body posture.
This gesture recognition system finds several applications in real life. The system implemented will work well for any home appliance where human gesture control is desired such as Smart TV, Smart computer etc.

It can also be used in medical applications, where we need to operate devices and instruments without making any physical contact with them.

Computer gaming is also a very important application for the hand gesture recognition system.
Conclusion and Future Scope

This thesis explored the implementation of visual data based, robust hand gesture recognition system. The platform used for the implementation of code is OpenCV using C languages.

The ability of our algorithm, to work without modelling of hand and machine learning, enables it to work straight away without training. We developed a novel approach for segmenting the hand, based on the constellation of fingertip points in space. A depth based histogram solves this problem.

This approach is robust and accurate irrespective of background clutter as well as varying lighting conditions.
Future Possibilities

Depth map of the Kinect is not of good quality, thus, the boundaries of the objects are not sharp. This problem can be overcome using some edge refining techniques.

This system can also be further extended for finger gesture recognition using some high resolution depth sensor.

A better quality 3D sensor can increase the range and performance of our system.

We can also develop a system for multi-user control. The users can be at the same depth or at different depths.


High Speed shape estimation

Goal
To compute the depth of a 3D object using structured light technique

Applications
Range sensor for navigation (Mars Exploration Rover), Biometrics (Face Recognition), Virtual Reality (Gesture Recognition) etc.

Figure 1: Depth Map - Color represents depth
Stereoscopic Systems

Two cameras which give disparity - difference in image locations of a point within the two cameras.

Depth from disparity by triangulation faces problem of correspondences.

\[ \text{Depth} = \frac{b \cdot f}{u_L - u_R} \]

**Figure 2:** Stereoscopic System
Structured Light Technique

One camera is replaced with light source which projects a known pattern. Based on the object shape these patterns deform. These deviations are used to calculate depth. Patterns projected assign unique codeword to a set of pixels to solve the correspondence problem.

Figure 3: (a) Structured Light Technique and (b) Binary Coding Technique
Calibration

Need to convert projector and camera images to a common reference frame for comparisons

Need to map 2D image coordinates to 3D reference coordinates

Figure 4: (a) Projected Frame, (b) Camera frame of illuminated background
Proposed Space Calibration

To find depth, transformation from cartesian space to camera coordinate system is needed.

Homography matrix defines the relation between two images, which are perspective transform versions of each other, in the form of a linear matrix.

But it gives the transformation between two planes and not two coordinate systems.

So to calculate the depth values along a single projected line, we compute homography matrices corresponding to two perpendicular planes.

One plane is $XY/|Z|=0$ and the other is $YZ$ plane.
Background plane is assumed to be XY plane.
For any line projected onto the background it’s camera image can be generated using XY homography matrix.
Presence of object in front of the background plane produces disparity.
Disparity - Deviation between the line locations in the camera image of illuminated object and in the image generated using XY homography.
Disparity is converted to depth using the YZ homography matrix.
XY Homography

Require a set of 4 points from both the projected and camera image as in Fig. 5

Projected image 4 points, $\mathbf{p}_{xy}$ - A, B, C and D
Camera image 4 points, $\mathbf{c}_{xy}$ - $A'$, $B'$, $C'$ and $D'$

Figure 5: (left) Projected image and (right) Camera image

$H_{xy} = \text{DLT}(\mathbf{p}_{xy}, \mathbf{c}_{xy})$
YZ Homography

For different planes in the real world, the homography matrices differ.

Hence Homography matrices have to be computed for planes parallel to every vertical stripe projected.

Two homography matrices - for two vertical stripes projected, left stripe and right stripe are computed and the rest are found by interpolation.

Four points are projected onto the background for which $z = 0$
YZ Homography

Left edge of the cuboid

Projected image 4 points, $\mathbf{P}_{yz_1}$ - A, B with $z=0$ and A and B with $z=10$ in Fig. 5

Camera image 4 points, $\mathbf{C}_{yz_1}$ - $A', B'$, $E'$ and $F'$ in Fig. 6

Figure 6:(left) Background image and (right) Foreground image

$$\mathbf{H}_{yz_1} = \text{DLT}(\mathbf{P}_{yz_1}, \mathbf{C}_{yz_1})$$
Right edge of the cuboid

Projected image 4 points, $P_{yz_2}$ - C, D with $z=0$ and C and D with $z=10$ in Fig. 5

Camera image 4 points, $C_{yz_2}$ - $C'$, $D'$, $G'$ and $H'$ in Fig. 6

$H_{yz_2} = \text{DLT}(P_{yz_2}, C_{yz_2})$

Linear interpolation for any vertical edge

$H_{yz} = \text{LinInterp}(H_{yz_1}, H_{yz_2})$
Hierarchical Orthogonal Codes

Hierarchical Orthogonal Codes (HOC)
- Provides a robust decision on pixel correspondence
- Reduces the code length while preserving the characteristics of the orthogonal code

N length code is divided into L layers and each layer includes H orthogonal codes recursively in Fig. 7
Codes in the HOC are not orthogonal but each layer has a set of orthogonal codes.

**Figure 7:** Hierarchical Orthogonal Codes with $H = 4$ and $L = 3$
Decoding in HOC

Decoding is based on the maximum likelihood of a code projected given the intensities of a pixel in the set of $H$ captured images of each layer in Fig. 8.

**Figure 8:** (a) intensity values at i-th pixel, (b) selecting the probable code for each layer, and (c) decoding of the correct address.
Self Occlusion test

Objects, based on their structure, may obstruct the light falling on themselves, resulting in errors in depth maps in Fig. 9. Shadow detection prior to depth computation.

Suggestion to the user to change camera or the projector position. Direction of translation can be deduced from the location of false locus/ shadow with respect to image center.

Figure 9: (left) Object with occlusion and (right) Object without occlusion.
Self Occlusion test

Works by finding the discontinuity loci in the image of an object illuminated with a set of white lines.
For a fully illuminated object, 3 starting and 3 ending discontinuity loci will be detected as in Fig. 10.
Background’s left boundary and object’s left and right boundaries.
Object’s left and right boundaries and background’s right boundary.

Figure 10: Discontinuity loci on a fully illuminated object.
Self Occlusion test

If the object has regions that are not illuminated, extra discontinuity loci appear.
If DiscontinuityLociNumber > 3 ⇒ Occlusion
Occlusion test with horizontal lines/vertical lines alone, detect shadowed regions, only when the shadow plane is not parallel to the lines as seen in Fig. 12
Test needs to be performed with both horizontal and vertical lines for detection of occlusion present in any direction.
The current algorithm traces curved lines too, as in Fig. 11, and hence this test is applicable on object of any shape.

Figure 11: Curved line tracing
Test failure cases

Figure 12: (a) Case where Horizontal stripes fail and Vertical stripes detect occlusion and (b) Case where Vertical stripes fail and Horizontal stripes detect occlusion
All the object pixels belonging to a segment share the same code. Depths are estimated only at the pixels on the segment boundaries and are interpolated later.

Correspondence between the edges in projected and camera images is attained, using HOC code values, as shown in Fig. 13.

**Figure 13:** (left) Edges with three different codes in the projected image and (right) Corresponding edges in the camera image of a box.
For a particular edge with HOC code $c$, let
- $p_{prj}$ be the pixel set in projected image
- $p'_{prj}$ be the XY transformed set
- $p_{cam}$ be the pixel set of its corresponding edge on object in camera view

$p'_{prj}$ and $p_{cam}$ are transformed to real world coordinates using YZ homography matrix computed for the plane containing this edge. Distance between them in Z direction is taken as depth. Repeated for edges of all code segments to generate a wire frame depth map. Interpolated to get smooth depth image.
Figure 14: (a) Actual Object, (b) Depth map for $L=4$ and $H=4$, (c) Depth map for $L=2$ and $H=8$ and (d) Depth map using binary code.
Figure 15: (a) Actual Object, (b) Depth map for $L=4$ and $H=4$, (c) Depth map for $L=2$ and $H=8$ and (d) Depth map using binary code
Figure 16: (a) Actual Object, (b) Depth map for $L=4$ and $H=4$, (c) Depth map for $L=2$ and $H=8$ and (d) Depth map using binary code
Figure 17: (a) Actual Object, (b) Depth map for $L=4$ and $H=4$, (c) Depth map for $L=2$ and $H=8$ and (d) Depth map using binary code
Accuracy

To calculate the accuracy, the depth of a uniform width box is calculated at multiple points both manually and through the proposed technique. Maximum error of 1mm which amounts to 1.3 percent of the object’s width. The RMSE value is 0.525mm.