Microsoft Kinect

Depth Sensing
Peter Sperl

C# developer at Anyline
Student at TU Wien (Medical Informatics)
Game developer hobbyist

Email: peter@anyline.io
Agenda

- Microsoft Kinect
  - About
  - Timeline
  - Components
  - Specifications
- Depth sensing
  - V1
    - Coded light
    - Triangulation
  - V2
    - Time of Flight
- Live Demo
• Motion sensing input device
• Natural user interface through voice and gestures
• Innovative use-cases
  – Gaming, Health Care, Fitness, Medicine, VR, AR, Robotics, etc.

• Sensor & SDK: For Windows PC and Xbox – Code written in C#, C++/CLI or VB.NET
Timeline

— 1\textsuperscript{st} generation:
  • 2010: Release for Xbox 360
  • 2011: Kinect for Windows SDK 1.x
  • 2012: Kinect for Windows V1

— 2\textsuperscript{nd} generation:
  • 2013: Release for Xbox One
  • 2014: Kinect for Windows V2 + SDK 2.0
  • 2015:
    — Kinect for Windows V2 product was discontinued, BUT:
    — Windows users simply use Kinect for Xbox One sensor + Kinect adapter
Components

- **Color sensor:**
  - common camera sensor to acquire RGB images
- **Depth camera:**
  - IR emitter + IR sensor use infrared light to acquire depth images
  - V1: Structured Light / Coded Light
  - V2: Time of Flight
- **Audio sensor:**
  - Microphone array to capture sound signal and sound position
# Kinect Specifications

<table>
<thead>
<tr>
<th>Feature</th>
<th>V1</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color stream</td>
<td>640x480 at 30FPS</td>
<td>1920x1280 at 30FPS</td>
</tr>
<tr>
<td>Depth stream</td>
<td>320x240</td>
<td>512x424</td>
</tr>
<tr>
<td>Infrared stream</td>
<td>---</td>
<td>512x424</td>
</tr>
<tr>
<td>Depth distance</td>
<td>0.4m (near mode) – 4m</td>
<td>0.4m – 4.5m</td>
</tr>
<tr>
<td>Horizontal field of view</td>
<td>57°</td>
<td>70°</td>
</tr>
<tr>
<td>Vertical field of view</td>
<td>43°</td>
<td>60°</td>
</tr>
<tr>
<td># defined skeleton joints</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td># bodies tracked</td>
<td>2 (+4 recognized)</td>
<td>6</td>
</tr>
<tr>
<td>Operating system</td>
<td>Windows 7 / 8</td>
<td>Windows 8</td>
</tr>
</tbody>
</table>
Kinect Specifications

V1

V2
Depth Sensing (V1)

1. IR Emitter (E) projects a known infrared light pattern ("coded light") into the scene
2. IR Sensor (S) retrieves reflected light from the environment
3. Depth map is computed using triangulation and light coding technology (by PrimeSense)
4. Depth data available via USB to the API
Coded Light
(also called structured light)

- Emitter projects pseudo-random pattern of light ("Speckles")
- Pattern is “coded”:
  - has to be unique per position so each point can be recognized in the pattern
Coded Light

- Light is distorted on the surface
- Pattern is unique in every position on the scene
- Allows to compute depth information through triangulation
Triangulation

- Used in stereo vision to reconstruct a point $P$ in 3D space, given two or more images
- Multiple image projections of the same scene yield depth information
- Positions, orientations, and optical properties of the image sources must be known
Problem:

- Triangulation requires two or more image sources, but Kinect has only one depth sensor.
Triangulation

• Trick: Kinect actually uses **two** images:
  
  – The image captured from the IR sensor
  – The “invisible” hardwired reference pattern

• Triangulation applicable because there is a distance between IR sensor and IR projector
Triangulation

• Simplified model:
  – Cameras are parallel along baseline B
  – Camera centers are at the same height
  – Cameras have same focal length f

• Goal: calculate disparity for each pixel
Triangulation

- Disparity $D$: Difference in image location of the same point $X$ when projected from different perspectives.

- Disparity is inversely proportional to depth.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$</td>
<td>Focal length</td>
</tr>
<tr>
<td>$D$</td>
<td>Measured disparity</td>
</tr>
<tr>
<td>$B$</td>
<td>Base length between emitter and camera</td>
</tr>
<tr>
<td>$z$</td>
<td>Distance to $X$</td>
</tr>
</tbody>
</table>

$$D = x - x' = \frac{(B \times f)}{z}$$

$$z = \frac{(B \times f)}{(x - x')}$$
Triangulation

- IR Pattern is correlated against the reference pattern
- Projected speckles:
  - if distance to the sensor is smaller or larger than on the reference plane: shifted in direction of the baseline between projector and perspective center of the IR camera
  - x-direction shifts are measured for each speckle
- Distance from each pixel to the sensor is retrieved from corresponding disparity

<table>
<thead>
<tr>
<th>$x_k, y_k$</th>
<th>Coordinates of point k</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_o, y_o$</td>
<td>Principal point offsets</td>
</tr>
<tr>
<td>$\sigma_x, \sigma_y$</td>
<td>Lens distortion corrections</td>
</tr>
<tr>
<td>$Z_0$</td>
<td>Distance between object and Sensor on reference plane</td>
</tr>
<tr>
<td>$Z_k$</td>
<td>Distance between point K and Sensor in object space</td>
</tr>
</tbody>
</table>

$$Z_k = \frac{Z_o}{1 + \frac{Z_o}{f b} d}$$

$$X_k = -\frac{Z_k}{f} (x_k - x_o + \delta x)$$

$$Y_k = -\frac{Z_k}{f} (y_k - y_o + \delta y)$$

![Diagram of triangulation](image)
Triangulation

• IR Sensor and IR Camera are *epipolar rectified*
  - Both have matching Field of View
  - Correspondence search reduced to 1D problem
  - If reflecting surface gets farther away, light from projector gets “larger” since each dot is a cone of light
  - Light expanding rate and camera pixel expansion are the same
• When object distance changes, dots appear to be shifted only in x-direction
• Tracking a dot’s x-coordinate yields the pixels observed depth
Depth Sensing (V2)

- 2\textsuperscript{nd} generation of Kinect uses Time of Flight to measure depth
- Indirect measurement of time it takes for laser pulses to a surface and back to the sensor

How is this possible?
Time of Flight

- Speed of light = 299,792,458 m / s
- 1 GHz CPU = 1*10^9 Hz = 1,000,000,000 Hz

-> Pulse of light travels ~ 29 cm within 1 CPU cycle

- IR Sensor:
  - divides 1 pixel in half
  - Half pixel is turned on and off extremely fast
    - On: Absorb photons of laser light
    - Off: Reject photons of laser light
  - Other half of the pixel does the same, but 180° phase shifted
Time of Flight

- Laser Illuminator (at the same time):
  - Pulse light source in phase of the first pixel-half
  - Only “on” when first pixel half is “on”

- Distance at 0 cm:
  - Light will be absorbed in the 1st half, nothing will happen in the second half

- Let’s move light source away for a bit:
  - Time of Flight will slightly increase, photons will return in the 2nd half and be rejected
  - Total amount of light absorbed will decrease slightly, total amount of light rejected will increase slightly
Time of Flight

- Moving source even farther away:
  - Absorbed amount decreases further, rejected amount increases further

- After several milliseconds exposure:
  - Two total amounts of photons absorbed by the two first halves are compared to the two total amounts of photons rejected by the two second halves
  - Ratio is important, not the total amount – surface can absorb light, but it affects both measurements equally

- Depth is measured by the round-trip distance
Time of Flight

- Moving even further away:
  - Photons might overshoot 1st and 2nd half and arrive in the next cycle
  - -> Ambiguity

- Ambiguity easily resolved by increasing pixel-half times, but then small depth changes can’t be measured

- Solution:
  - One low resolution measurement as estimate
  - One high precision measurement, using estimate to eliminate ambiguities
Live Demo
Thank you!

And thanks PRIP for lending me the Kinect for Xbox One sensor 😊

Pattern Recognition Image Processing Group @ TU Wien
• **Sources:**
  - [Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications](https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect16.pdf)
  - [http://www.visionrt.com/content/core-technology](http://www.visionrt.com/content/core-technology)

• **Images:**
  - [http://vision.ia.ac.cn/Students/gzp/kinect1.html](http://vision.ia.ac.cn/Students/gzp/kinect1.html)
  - [http://2.bp.blogspot.com/-f4tcmi3LrDE/UZ0uY5f2p6I/AAAAAAAAAOY/8FQIuBG8suA/s1600/new-kinect-internal.png](http://2.bp.blogspot.com/-f4tcmi3LrDE/UZ0uY5f2p6I/AAAAAAAAAOY/8FQIuBG8suA/s1600/new-kinect-internal.png)

• **Code Example:**