Range Imaging Through Triangulation

Obviously, this is a very slow process and not suitable for dynamic scenes.
To speed things up, we can use a laser that projects a vertical line of light onto the scene.
This laser rotates around its vertical axis and thereby moves the vertical line of light across the scene.
Since only the horizontal positions of points vary and give us depth information, the vertical order of points is preserved.
This allows us to compute the depth of each point along the vertical line without ambiguities.

Range Imaging Through Triangulation

Although this method is faster, it still requires a complete horizontal scan before a depth image is complete.
Maybe we should use a pattern of many vertical lines that only needs to be shifted by the distance between neighboring lines?
The disadvantage of this idea is that we could confuse points in different vertical lines, i.e., associate points with incorrect projection angles.
However, we can overcome this problem by taking multiple images of the same scene with the pattern in the same position.
In each picture, a different subset of lines is projected.

The Microsoft Kinect V1 System

![Diagram of the Microsoft Kinect V1 System]

The Microsoft Kinect V1 System uses an IR (Infrared) camera to capture depth images. The IR camera emits infrared light, which is reflected back from objects in the scene. By analyzing the time it takes for the light to return and the intensity of the reflected light, the system can determine the distance to objects in the scene.

Obviously, with this technique we can encode up to \((n - 1)\) lines using \(\log_2(n)\) images. Therefore, this method is more efficient than the single-line scanning technique.
The Microsoft Kinect V1 System

The Microsoft Kinect V2 System

The Kinect V2 uses a time-of-flight camera to measure depth.
It contains a broad IR illuminator whose light is intensity modulated:

The reflected light is projected through a lens onto an array of IR light sensors.
For each sensor element, the phase shift between the reflected and the currently emitted light is computed.

The reflecting light is projected through a lens onto an array of IR light sensors.
For each sensor element, the phase shift between the reflected and the currently emitted light is computed.

Motion analysis

Motion analysis is dealing with three main groups of motion-related problems:
- Motion detection
- Moving object detection and location.
- Derivation of 3D object properties.

Motion analysis and object tracking combine two separate but inter-related components:
- Localization and representation of the object of interest (target).
- Trajectory filtering and data association.

One or the other may be more important based on the nature of the motion application.

Motion analysis

Figure 16.1 Object motion assumptions. (a) Minimum velocity (the shaded circle represents the area of possible object locations). (b) Fixed acceleration (the shaded circle represents the area of possible object locations at time $t_0$). (c) Common motion and initial correspondence (rigid object).
**Differential Motion Analysis**

A simple method for motion detection is the subtraction of two or more images in a given image sequence.

Usually, this method results in a difference image $d(i, j)$, in which non-zero values indicate areas with motion.

For given images $f_1$ and $f_2$, $d(i, j)$ can be computed as follows:

$$d(i, j) = \begin{cases} 
0 & \text{if } |f_1(i, j) - f_2(i, j)| \leq \varepsilon \\
1 & \text{otherwise}
\end{cases}$$

---

**Difference Pictures**

Another example of a difference picture that indicates the motion of objects ($\varepsilon = 25$).

---

**Differential Motion Analysis**

In order to determine the direction of motion, we can compute the cumulative difference image for a sequence $f_1, \ldots, f_n$ of more than two images:

$$d_{cum}(i, j) = \sum_{k=2}^{n} a_k |f_k(i, j) - f_1(i, j)|$$

Here, $f_1$ is used as the reference image, and the weight coefficients $a_k$ can be used to give greater weight to more recent frames and thereby highlight the current object positions.
Cumulative Difference Image

\[ d_{cum}(i,j) = \sum_{k=2}^{n} a_k | f_k(i,j) - f_{k+1}(i,j) | \]

Example: Sequence of 4 images:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a_2 = 1</td>
<td>a_3 = 2</td>
<td>a_4 = 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Result:

<table>
<thead>
<tr>
<th>0</th>
<th>7</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Differential Motion Analysis

Generally speaking, while differential motion analysis is well-suited for motion detection, it is not ideal for the analysis of motion characteristics.

Optical Flow

Optical flow reflects the image changes due to motion during a time interval \( dt \) which must be short enough to guarantee small inter-frame motion changes.

The optical flow field is the velocity field that represents the three-dimensional motion of object points across a two-dimensional image.

Optical flow computation is based on two assumptions:

- The observed brightness of any object point is constant over time.
- Nearby points in the image plane move in a similar manner (the velocity smoothness constraint).

Optical Flow

The basic idea underlying most algorithms for optical flow computation:

- Regard image sequence as three-dimensional \((x, y, t)\) space
- Determine \(x\)- and \(y\)-slopes of equal-brightness pixels along \(t\)-axis

The computation of actual 3D gradients is usually quite complex and requires substantial computational power for real-time applications.
Optical Flow

Instead of using the gradient methods, one can simply determine those straight lines with a minimum of variation (standard deviation) in intensity along them:

Optical flow

Optical flow computation will be in error if the constant brightness and velocity smoothness assumptions are violated. In real imagery, their violation is quite common. Typically, the optical flow changes dramatically in highly textured regions, around moving boundaries, at depth discontinuities, etc. Resulting errors propagate across the entire optical flow solution.

Block-Matching Motion Estimation

A simple method for deriving optical flow vectors is block matching. The current video frame is divided into a large number of squares (blocks). For each block, find that same-sized area in the following frame(s) with the greatest intensity correlation to it. The spatiotemporal offset between the original block and its best matches in the following frame(s) indicates likely motion vectors. The search can be limited by imposing a maximum velocity threshold.

Feature Point Correspondence

The idea is to find significant points (interest points, feature points) in all images of the sequence—points least similar to their surroundings, representing object corners, borders, or any other characteristic features in an image that can be tracked over time. Basically the same measures as for stereo matching can be used. Point detection is followed by a matching procedure, which looks for correspondences between these points in time. The main difference to stereo matching is that now we cannot simply search along an epipolar line, but the search area is defined by our motion assumptions. The process results in a sparse velocity field. Motion detection based on correspondence works even for relatively long interframe time intervals.