3D reconstruction with depth image fusion

Szirmay-Kalos László

Depth (range) cameras

Depth compositing (Zinemath)

Application: limitations of compositing

Chroma keying

Augmented reality

Compositing can be based on color:
• Fixed order
• No shadows
• No reflections, refractions, cross illumination
3D reconstruction of a point

\[
\mathbf{R} = \begin{bmatrix}
  i_x' & i_y' & k_x' \\
  i_x' & i_y' & k_y' \\
  i_x' & i_y' & k_z'
\end{bmatrix}
\]

\[
\mathbf{p}_w = \mathbf{R}_w \mathbf{p}_c + \mathbf{t}
\]

\[
\mathbf{f}(X-c_x)s_x, f(Y-c_y)s_y, z_c, t = \mathbf{d}(X,Y)z_c + t
\]

Back projection

Stereo vision and active light

\[
p_w = d_1z_1 + t_1
\]

\[
p_w = d_2z_2 + t_2
\]

\[
p_w = d_3z_3 + t_3
\]

Time of flight sensors

Pulsed modulation:
Accurate time measurement
Expensive

Continuous modulation:
Periodic distance

Temporal noise filtering and back projection for static camera
**3D triangle mesh: height field**

Topology is a height field

---

**Linear and Bilateral filtering**

Linear filter:

\[
\tilde{D}(p) = \frac{\sum w(|o|)D(p+o)}{\sum w(|o|)} - \frac{\sum w(|o|)D(p-o)}{\sum w(|o|)}
\]

Bilateral filter:

\[
\tilde{D}(p) = \frac{\sum w(|o|)W(|D(p+o) - D(p)|)D(p+o)}{\sum W(|o|)W(|D(p+o) - D(p)|)}
\]

Generalization:

\[
\tilde{D}(p) = \frac{\sum r(o,p+o\ast)D(p+o)}{\sum r(o,p+o\ast)}
\]

\[
r(o,p+o\ast) = w(|o|)w(|D(p+o) - D(p)|)w(|I(p+o) - I(p)|)
\]

---

**Moving objects: optical flow**

Classical optical flow constraint:

\[
E_c(p,v) = ||I(p,t) - I(p+v,t+1)||
\]

Gradient constancy:

\[
E_g(p,v) = \left| \frac{d}{dp}I(p,t) - \frac{d}{dp}I(p+v,t+1) \right|
\]

Smoothness:

\[
E_s(p,v) = \left| \frac{\partial v_x}{\partial x} \right|^2 + \left| \frac{\partial v_y}{\partial y} \right|^2 + \left| \frac{\partial v_y}{\partial x} + \frac{\partial v_x}{\partial y} \right|^2
\]

Energy functional:

\[
E(v) = \int E_c(p,v) + \alpha E_g(p,v) + \beta E_s(p,v) dp
\]

---

**Spatio-Temporal filtering**

\[
E_c(p,q,v) = ||I(p) - I(q)||
\]

\[
w_c(||I(p) - I(q)||)
\]

\[
w_d(||D(p) - D(q)||)
\]

\[
w_s(||p^* - q^*||)
\]

\[
w_d(q,q^*)
\]
**System**

- RGB camera
- Depth camera
- GLSL shaders
- Lens distortion compensation
- Optical flow
- R.I. reprojection + upsampling
- Spatial-temporal filtering
- Filtered point cloud
- Rendering with OpenGL

**Dynamic camera, static scene**

- Point cloud
- Fusion
- Projection
- Object

Problems:
- in different images the camera changes
- camera tracking based on static objects
- in different frames different points are visible

We need to maintain surface information between points

Solution (Curless/Levoy):
- Scene is represented by an emerging distance field

**Results**

- Distance field (signed, truncated)

- Voxels
- Object
Ray marching on GPU

```cpp
for(t = 0; t < exit; t += dt) {
    float3 q = eye + raydir * t;
    if (tex3d(volume, q) < 0) {
        float3 normal = float3( tex3d(volume, q + float3(1/RES,0,0)) -
                               tex3d(volume, q - float3(1/RES,0,0)),
                               tex3d(volume, q + float3(0,1/RES,0)) -
                               tex3d(volume, q - float3(0,1/RES,0)),
                               tex3d(volume, q + float3(0,0,1/RES)) -
                               tex3d(volume, q - float3(0,0,1/RES)))!
        normal = normalize(normal);
        color = lightint * kd * max(dot(lightdir, normal), 0);
        distance = t;
        return;
    }
    color = float3(0,0,0);
    distance = -1;
    return;
}
```

Surface generation: marching cubes

Volumetric fusion

<table>
<thead>
<tr>
<th>0</th>
<th>2</th>
<th>14</th>
<th>255</th>
</tr>
</thead>
<tbody>
<tr>
<td>0;</td>
<td>0-1; 0-3; 2-6; 0-3; 3-7; 2-6</td>
<td>0;</td>
<td>Case: 00001110_2 = 14</td>
</tr>
</tbody>
</table>
Camera tracking: Iterative Closest Point

\[ E(R, t) = \frac{1}{n} \sum_{i=1}^{n} \| p_i - R \cdot q_i - t \| \rightarrow \min \]

Projective data association

Backprojection of current image

Zero crossings of the 3D distance field

System

- Volumetric model: 128 x 128 x 128
- Image resolution: 640 x 480
- Speed: 30 FPS
- Hardware: NVIDIA 690 GT, CUDA

Result on synthetic data

Additional filtering of distance fields
Additional filtering of distance field

Implementation and Conclusions

• With GPU real-time 3D reconstruction is possible
• Height field representation:
  – Simplified implementation: http://3d-scene.org/
• Volumetric reconstruction:
  – Public domain simplified implementation: PCL library
  – Commercial: KinectFusion