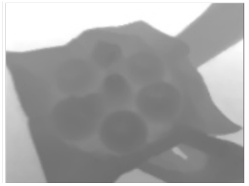
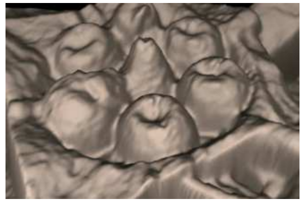
BME IIT CG

3D reconstruction with depth image fusion

Szirmay-Kalos László

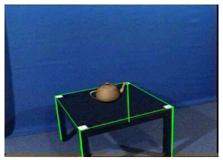
Depth (range) cameras



Application: limitations of compositing

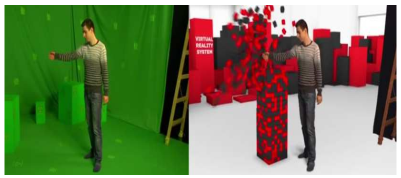
Chroma keying

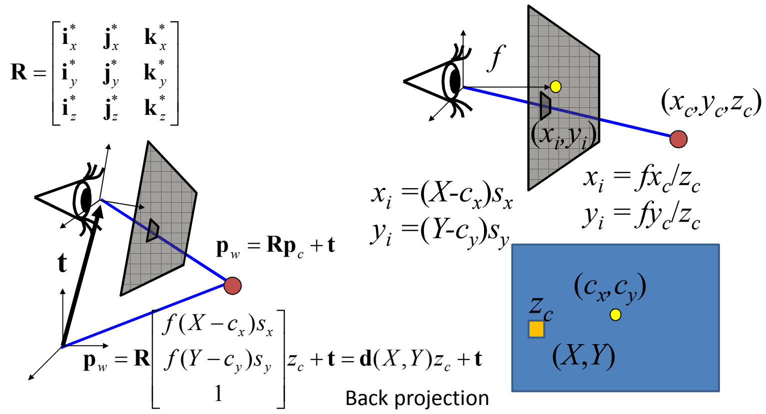
Augmented reality


Compositing can be based on color:
- Fixed order
- No shadows
- No reflections, refractions, cross illumination

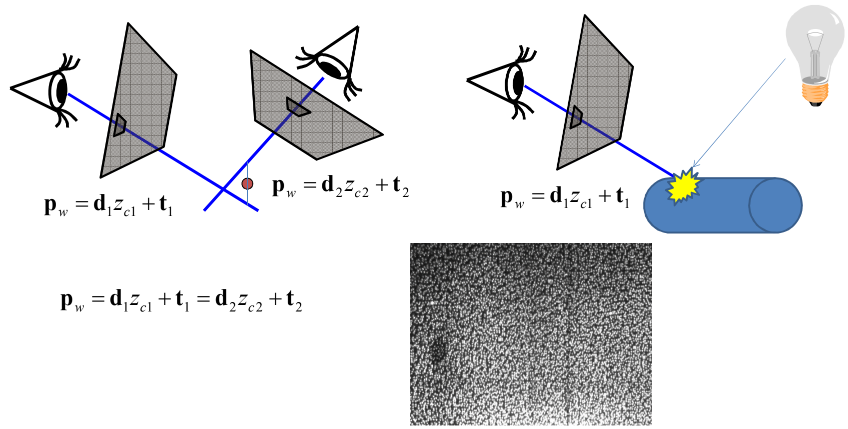
Depth compositing (Zinemath)

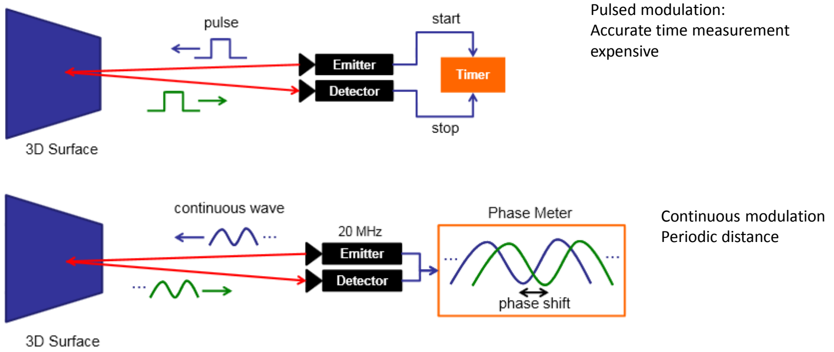
3D reconstruction of a point



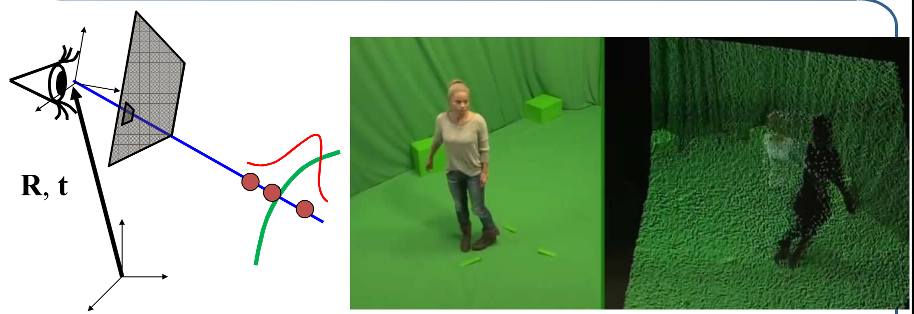
Stereo vision and active light



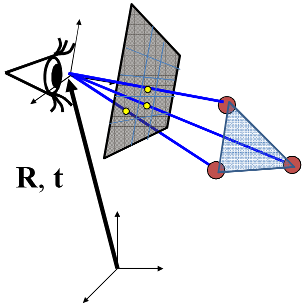
Time of flight sensors

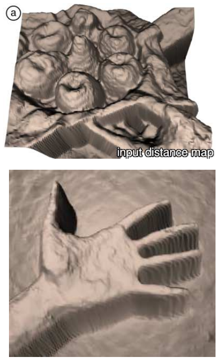


Temporal noise filtering and back projection for static camera



3D triangle mesh: height field



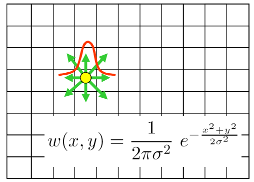


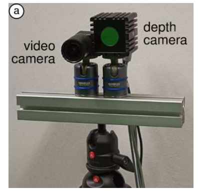
Topology is a height field

Linear and Bilateral filtering

Linear filter:

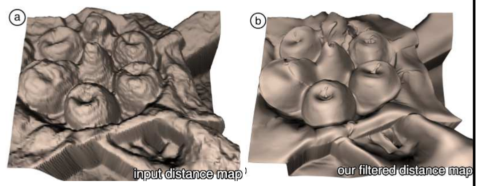
$$\tilde{D}(\mathbf{p}) = \frac{\sum w(|\mathbf{o}|) D(\mathbf{p}+\mathbf{o})}{\sum w(|\mathbf{o}|)} = \sum w(|\mathbf{o}|) D(\mathbf{p}+\mathbf{o})$$





Bilateral filter:

$$\tilde{D}(\mathbf{p}) = \frac{\sum w_s(|\mathbf{o}|) w_i(|D(\mathbf{p}+\mathbf{o}) - D(\mathbf{p})|) D(\mathbf{p}+\mathbf{o})}{\sum w_s(|\mathbf{o}|) w_i(|D(\mathbf{p}+\mathbf{o}) - D(\mathbf{p})|)}$$



Generalization:

$$\tilde{D}(\mathbf{p}) = \frac{\sum r(\mathbf{o},\mathbf{p}+\mathbf{o},\bullet) D(\mathbf{p}+\mathbf{o})}{\sum r(\mathbf{o},\mathbf{p}+\mathbf{o},\bullet)}$$

$$r(\mathbf{o},\mathbf{p}+\mathbf{o},\bullet) = w_s(|\mathbf{o}|) w_d(|D(\mathbf{p}+\mathbf{o}) - D(\mathbf{p})|) w_c(|\mathbf{I}(\mathbf{p}+\mathbf{o}) - \mathbf{I}(\mathbf{p})|)$$

Moving objects: optical flow

Classical optical flow constraint:

$$E_c(\mathbf{p},\mathbf{v}) = |\mathbf{I}(\mathbf{p},t) - \mathbf{I}(\mathbf{p}+\mathbf{v},t+1)|$$

Gradient constancy:

$$E_g(\mathbf{p},\mathbf{v}) = \left|\frac{\mathrm{d}}{\mathrm{d}\mathbf{p}}\mathbf{I}(\mathbf{p},t) - \frac{\mathrm{d}}{\mathrm{d}\mathbf{p}}\mathbf{I}(\mathbf{p}+\mathbf{v},t+1)\right|$$

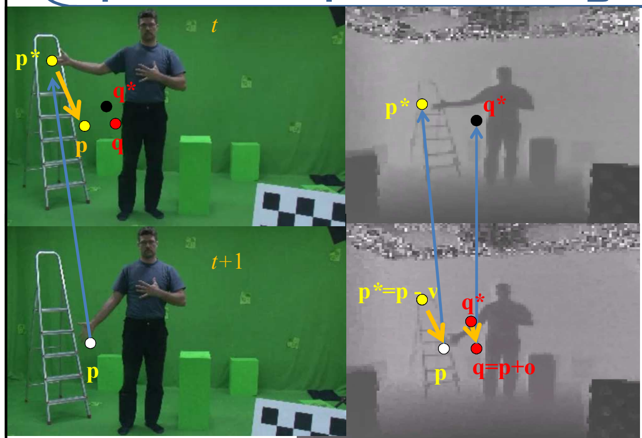
Smoothness:

$$E_s(\mathbf{p},\mathbf{v}) = \left|\frac{\partial \mathbf{v}_x}{\partial x}, \frac{\partial \mathbf{v}_x}{\partial y}, \frac{\partial \mathbf{v}_x}{\partial t}\right|^2 + \left|\frac{\partial \mathbf{v}_y}{\partial x}, \frac{\partial \mathbf{v}_y}{\partial y}, \frac{\partial \mathbf{v}_y}{\partial t}\right|^2$$

Energy functional:

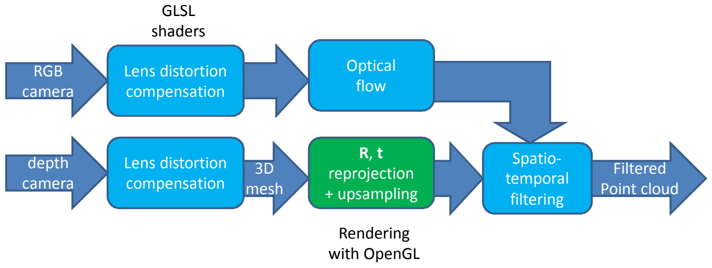
$$E(\mathbf{v}) = \int E_c(\mathbf{p},\mathbf{v}) + \alpha E_g(\mathbf{p},\mathbf{v}) + \beta E_s(\mathbf{p},\mathbf{v})\mathrm{d}\mathbf{p}$$

Spatio-Temporal filtering



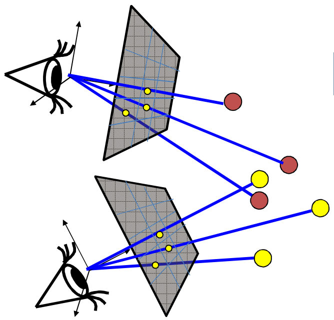
$$r(\mathbf{p},\mathbf{q},\mathbf{p}^*,\mathbf{q}^*) = w_c(|\mathbf{I}(\mathbf{p}) - \mathbf{I}(\mathbf{q}^*)|)\, w_d(|D(\mathbf{p}) - D(\mathbf{q}^*)|)\, w_s(|\mathbf{p}^*,\mathbf{q}^*|)\, w_d(\mathbf{q},\mathbf{q}^*)$$

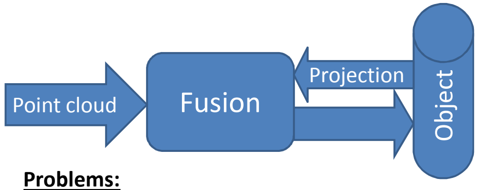
System



Results

Dynamic camera, static scene





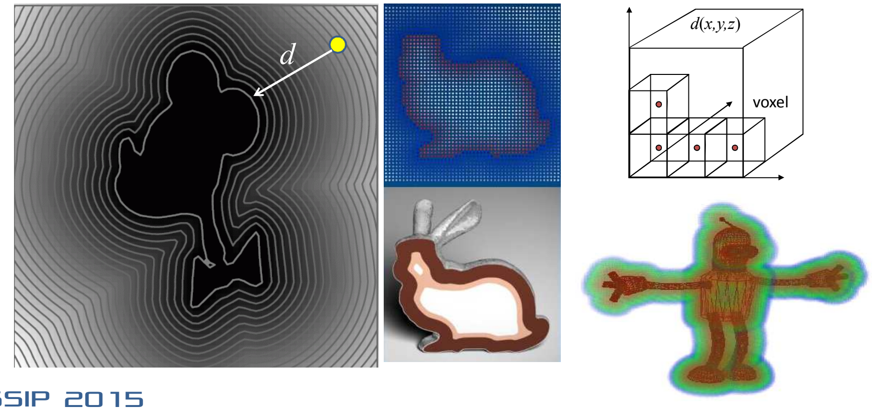
Problems:

- 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**

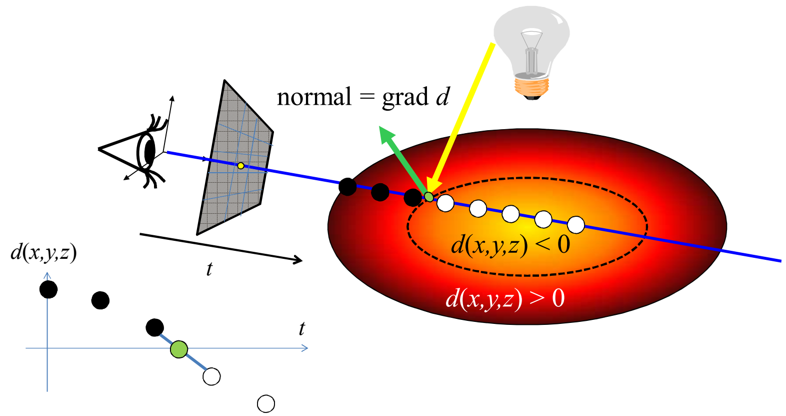
Solution (Curless/Levoy):

- Scene is represented by an emerging distance field

Distance field (signed, truncated)



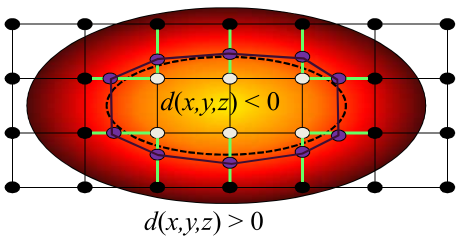
Rendering distance fields: ray marching

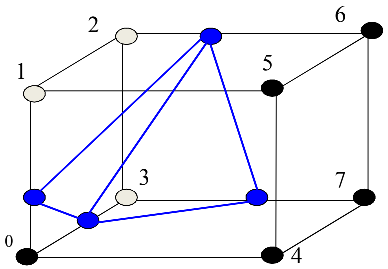


Ray marching on GPU

```
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

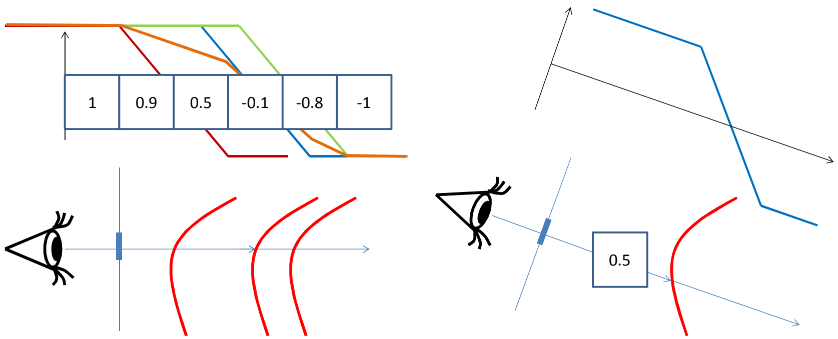




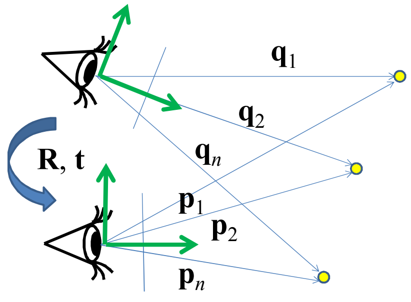
Case: $00001110_2 = 14$

0	0;	
14	2; 0-1; 0-3; 2-6; 0-3; 3-7; 2-6	14
255	0;	

Volumetric fusion

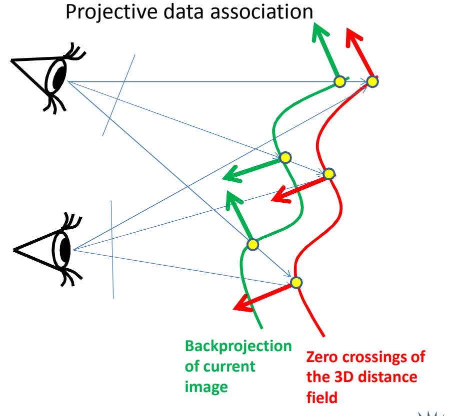


Camera tracking: Iterative Closest Point

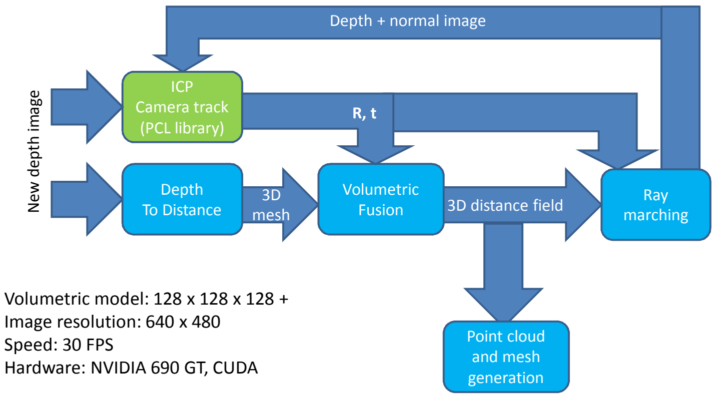


$$E(\mathbf{R},\mathbf{t}) = \frac{1}{n}\sum_i \|\mathbf{p}_i - \mathbf{R}\cdot\mathbf{q}_i - \mathbf{t}\| \rightarrow \min$$

Projective data association

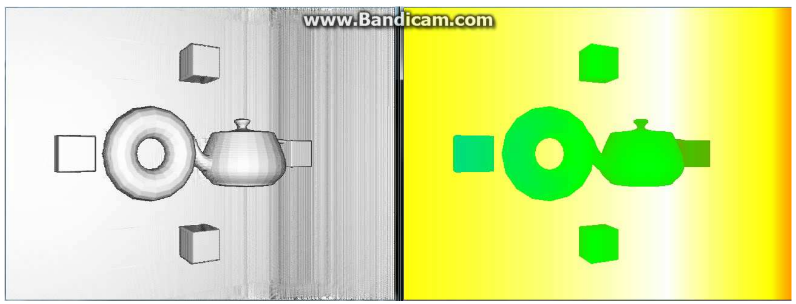


System

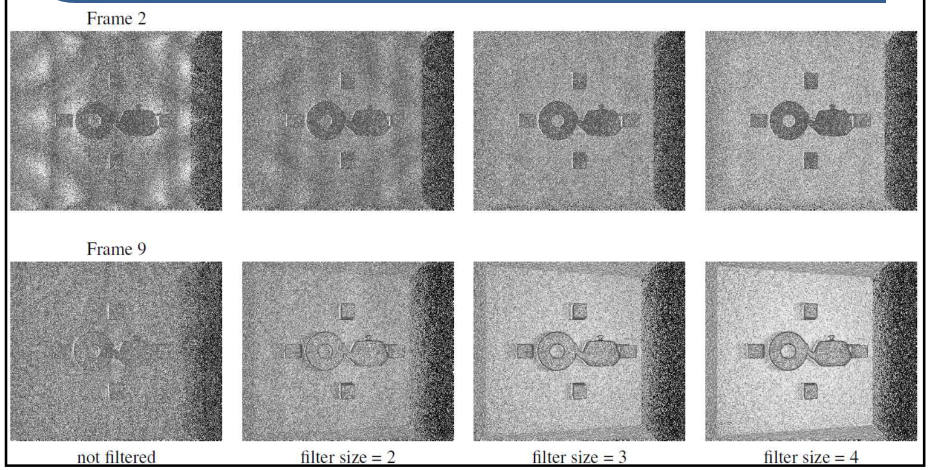


- Volumetric model: 128 x 128 x 128 +
- Image resolution: 640 x 480
- Speed: 30 FPS
- Hardware: NVIDIA 690 GT, CUDA

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