Narrow Band Level set on GPU

Chunliang Wang
CMIV, Linköping University
Image Segmentation
Contour based Segmentation

\[ V(s) = [x_i, y_i]^T \]

\[
\min_v E = \int_0^1 E_{int}(v(s)) + E_{ext}(v(s)) + E_{const}(v(s)) \, ds
\]
Internal Force - Curvature
External Force-Intensity
Explicit Contour
Implicit Contour: Level-set

Explicit Contour->Level Set

Level set in Physics
Level set For Image segmentation

\[ \frac{\partial \phi}{\partial t} = -|\nabla \phi| \left[ \alpha D(I) + (1 - \alpha) \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right] \]
What is done in each iteration

\[
\frac{\partial \phi}{\partial t} = -|\nabla \phi| \left[ \alpha D(I) + (1 - \alpha) \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right]
\]

- Calculate external speed (intensity)
- Calculate internal speed (curvature)
- Calculate local gradient
- Calculate the local step length (sum of speeds*gradient* \( dt \))
- Add the step length to the level set function
void updatephi(float *d_phi, float *d_phi1, float *d_D, int imageH, int imageW)
{
    int c = blockIdx.x * blockDim.x + threadIdx.x;
    int r = blockIdx.y * blockDim.y + threadIdx.y;
    int ind = r*imageW + c;
    if(ind<imageW*imageH)
    {
        float dx, dy, dxplus, dxminus, dyplus, dyminus;
        float gradphi, x, y, xplus, xminus, yplus, yminus;
        float dxplusy, dyplusx, dxminusy, dyminusx;
        float dxplus, dyplus, dxminus, dyminus, dxplusydxplusx, dxminusydxminusx, dyplusxdyplusy, dyminusxdyminusy;
        float dxplusydxplusx = dxplusy * dxplusx;
        float dxminusydxminusx = dxminusy * dxminusx;
        float dyplusxdyplusy = dyplusx * dyplusy;
        float dyminusxdyminusy = dyminusx * dyminusy;
        float gradphi = sqrt(dxplusydxplusx + dyplusxdyplusy);
        float gradphi1 = sqrt(dxminusydxminusx + dyminusxdyminusy);
        float F = ((1 - ALPHA) * d_D[ind]) + (1 - ALPHA) * curvature;
        if(F > 0) if(gradphi > gradphi1) d_phi1[ind] = d_phi[ind] + (F * gradphi);
        else if(gradphi < gradphi1) d_phi[ind] = d_phi[ind] + (F * gradphi1);
        else d_phi[ind] = d_phi[ind] + (F * gradphi1);
    }
}
GPU shared mem vs. CPU

- Test image 512*512 (30 time faster)
Narrow Band level set

26s on CPU

Drawbacks of Narrow band on GPU

- No shared memory
- No coalescing
- List operation
Generating Active Points list using CUDPP on GPU

Variable Output Per Thread: Compact

- Remove null elements
- Example: collision detection

Ultra-Narrow Band

Sparse field level set

13 s on CPU

Generate list from list

<table>
<thead>
<tr>
<th>index1</th>
<th>index2</th>
<th>index3</th>
<th>index4</th>
<th>index5</th>
</tr>
</thead>
<tbody>
<tr>
<td>neigbor1</td>
<td>neigbor1</td>
<td>neigbor1</td>
<td>neigbor1</td>
<td>neigbor1</td>
</tr>
<tr>
<td>neigbor2</td>
<td>neigbor2</td>
<td>neigbor2</td>
<td>neigbor2</td>
<td>neigbor2</td>
</tr>
</tbody>
</table>

- Index2’s neighbor
- Index1’s neighbor
- Index1’s neighbor
- Index1’s neighbor
- Index2’s neighbor
Lazy narrow band levelset

0.8 s on CPU

Lazy narrow band levelset

On 3D

- 256*256*256 dataset

<table>
<thead>
<tr>
<th></th>
<th>gpu</th>
<th>cpu</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td></td>
<td>360-</td>
</tr>
<tr>
<td>sparse field</td>
<td>70</td>
<td>450</td>
</tr>
<tr>
<td>lazy sparse field</td>
<td>141</td>
<td>90</td>
</tr>
</tbody>
</table>

1. I haven’t be able to make my 3D version works on GPU, the 14s record is from Roberts’ paper.

Thank You!