1. Abstract
In the context of video segmentation with depth sensor, prior work [1] maps the Metropolis algorithm, a simulated annealing-based segmentation routine, onto an Nvidia Graphics Processing Unit (GPU) and achieves real-time performance for 320x256 video sequences. However, that work utilizes depth information in a very limited manner. Our work extends the GPU-based method to use depth information during segmentation and shows the improved segmentation quality over the prior work.

We introduce a scaling factor for amplifying the interaction strength between two spatially neighboring pixels and increasing the clarity of boundaries. This allows us to reduce the number of required Metropolis iterations by over 50% with the drawback of over-segmentation. We evaluate two design choices to overcome this problem.

First, we pre-process the frames with Bilateral filter instead of Gaussian filter, and show its effectiveness in terms of reducing the difference between similar colors. Second, we incorporate depth information into the perceived color difference calculations between two pixels, and show that the interaction strengths between neighboring pixels can be more accurately modeled by incorporating depth information. Both approaches help improve the quality of the segmentation, and the reduction in Metropolis iterations helps improve the throughput from 29 fps to 34 fps for 320x256 video sequences. A link to the video outputs have been provided in references [2].

2. Problem Statement
• Image Segmentation is the process of partitioning a group of pixels having common characteristics into multiple sets, each set usually representing whole objects.
• Traditionally, segmentation has been carried out based only on visual cues by the color information conveyed through the pixels. This has a few drawbacks as in the scene below where a white paper plate and white kettle, placed in front of a white wall, are erroneously segmented as parts of the wall as shown in Fig. 1(a) & 1(b).
• Using a depth sensor, it would be possible to detect objects distinctly even when similarly colored and occluded as shown in Fig. 1(c) & 1(d).
• However, processing the sheer volume of pixel and depth data generated, especially in video sequences, is a bottleneck, unsuitable for many applications with real-time requirements.
• In this work, we address the problem of achieving high quality video segmentation in real-time.

3. Technology
• Recent technological advances make it possible to realize a real-time video segmentation system. Availability of capable but inexpensive depth sensors such as the Microsoft Kinect and ASUS Xtion Pro provide both color and depth information via an RGB camera and an infrared (IR) camera respectively.
• The computational power provided by the General Purpose Graphics Processing Units (GPUGPUs) are suited for data intensive applications with a high degree of inherent parallelism.
• For this project, we integrate NVIDIA’s GTX480 & the Microsoft Kinect with a Xeon x5675 CPU to implement our real-time video segmentation system.

4. Segmentation flow
• The core segmentation algorithm [1] is based on the Metropolis procedure. The pixels of a frame form a system similar to particles in a crystal lattice. The lowest energy state of such a system corresponds to a fully segmented frame.

4.1. Segmentation Algorithm on the GPU
• This procedure is used to process Fig. 4(a) to give 4(b) and Fig. 4(e) to give Fig. 4(f). The pixels of a frame form a system similar to particles in a crystal lattice.

4.2. Segmentation Algorithm on the GPU
• The algorithm benefits from a GPU since each pixel’s energy is based on interactions with neighboring pixels (Fig. 2). These interactions are independent from one another and therefore can be carried out in parallel. Interaction strengths of pixels at different depths are explicitly weakened.

5. Optimization strategies
• A serial and GPU-based version of the algorithm described earlier [1], were implemented. A comparison of their running times for one frame is listed on the table to the left.

5.1. Execution Times 
<table>
<thead>
<tr>
<th>Task</th>
<th>Serial*</th>
<th>CPU-GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metropolis Core</td>
<td>3.199</td>
<td>0.013</td>
</tr>
<tr>
<td>Auxiliary Tasks</td>
<td>0.006</td>
<td>0.006</td>
</tr>
<tr>
<td>Total</td>
<td>3.230</td>
<td>0.019</td>
</tr>
<tr>
<td>Speed-up</td>
<td>60x</td>
<td></td>
</tr>
</tbody>
</table>

5.2. Reduction in Metropolis iterations
• Applying the algorithm as is to each frame of the video achieves a throughput of 18.6 frames per second (fps) which is not good enough for most real-time applications.

5.3. Bilateral filter for noise removal
• Bilateral filtering frames before processing is a good way of removing artificial boundaries introduced by lighting variations and also due to the reduction in metropolis iterations.

5.4. Depth enhanced coupling matrix
• A bilateral filter is able to get rid of the vertical dark striations from the door which would otherwise cause poor segmentation.
• Since this filter is computationally intensive, throughput reduces to 34 fps. But quality of segmentation improves over the original segmentation flow as shown in Fig 6(d) and 6(e).

5.5. Bilateral filter for noise removal
• A serial and a GPU-based version of the algorithm described earlier [1], were implemented. A comparison of their running times for one frame is listed on the table to the left.

6. References
• Whenever the cases below are encountered, the interaction strengths of the participating pixels are set to the maximum possible value, thereby classifying them in the same segment.

6.1. Reduction in Metropolis iterations
• Same depth, same color
• Same depth, slightly different colors
• Slightly different depths, same colors
• Slightly different depths, slightly different colors
• This approach gives the best quality of segmentation with a throughput of 34 fps. Below is a comparison of two similar scenes with the original segmentation flow output using depth (Fig. 7(a)) and our depth enhanced coupling matrix segmentation flow output (Fig. 7(b)). The video showing the depth enhanced segmentation output can be found at [2].

6.2. Bilateral filter for noise removal