Real-Time GPU Based Video Segmentation with Depth Information

THE UNIVERSITY OF ARIZONA
ece electrical and computer engineering

Nilangshu Bidyanta, Ali Akoglu
nbidyanta@email.arizona.edu, akoglu@email.arizona.edu

NVIDIA CUDA TEACHING CENTER

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.

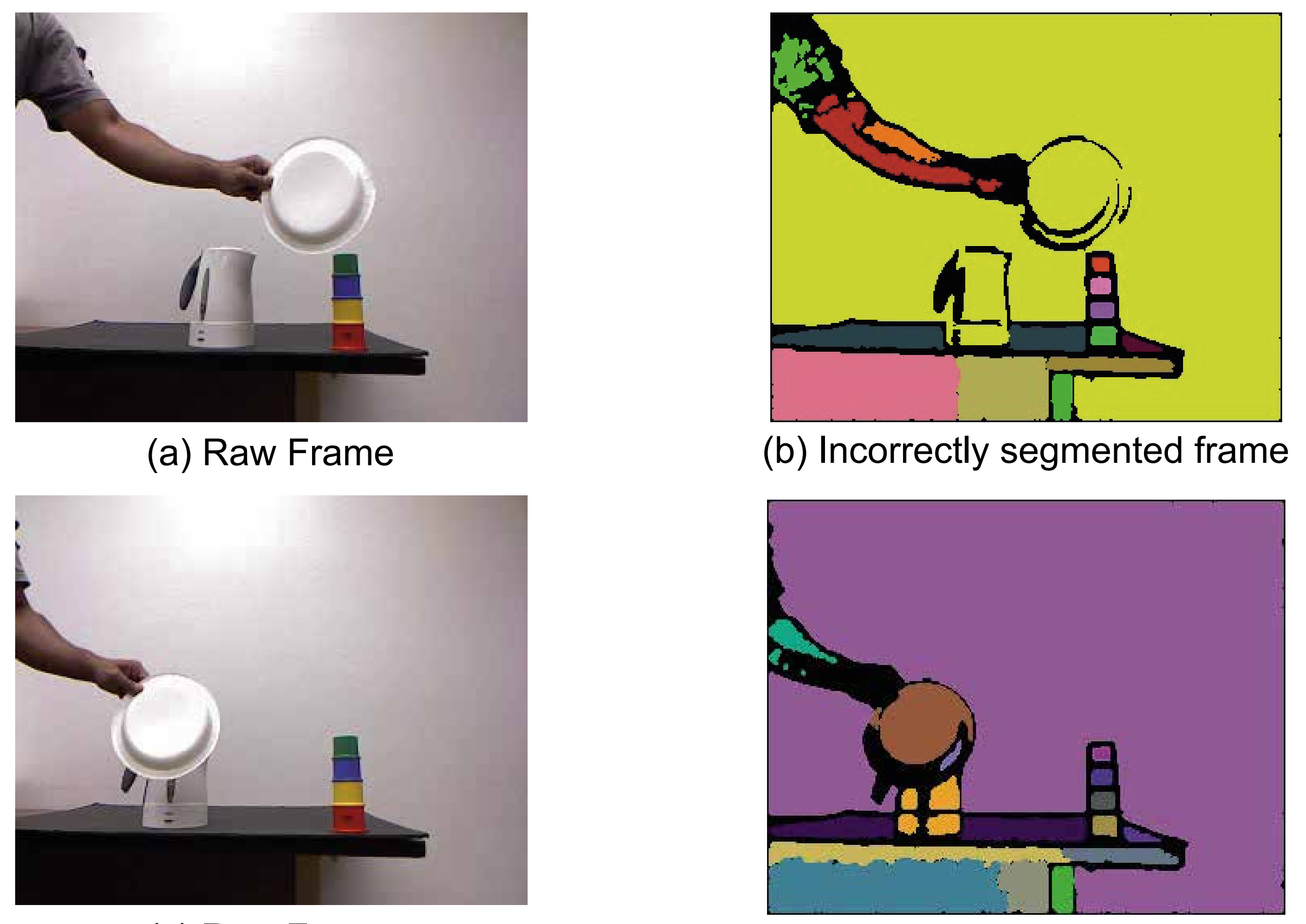
We introduce a scaling factor for amplifying the interaction strength between two spatially neighboring pixels and increasing the clarity of borderlines. This allows us to reduce the number of required Metropolis iterations by over 50% with the drawback of over-segmentation. We evaluate two design choices 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 while wall, are erroneously segmented as parts of the wall as shown in *Fig. 1(a) & 1(b)*.


(a) Raw Frame (b) Incorrectly segmented frame
(c) Raw Frame (d) Correctly segmented frame using depth
Fig. 1

• 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 (GPGPUs) 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.

3. Segmentation algorithm on the GPU

• 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.

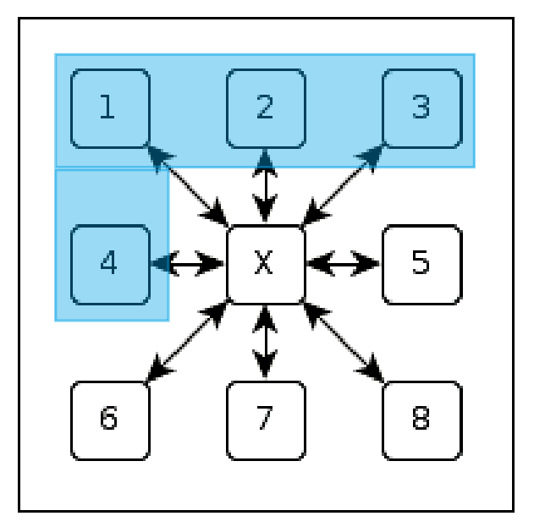


Fig. 2

• 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.

• 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.***

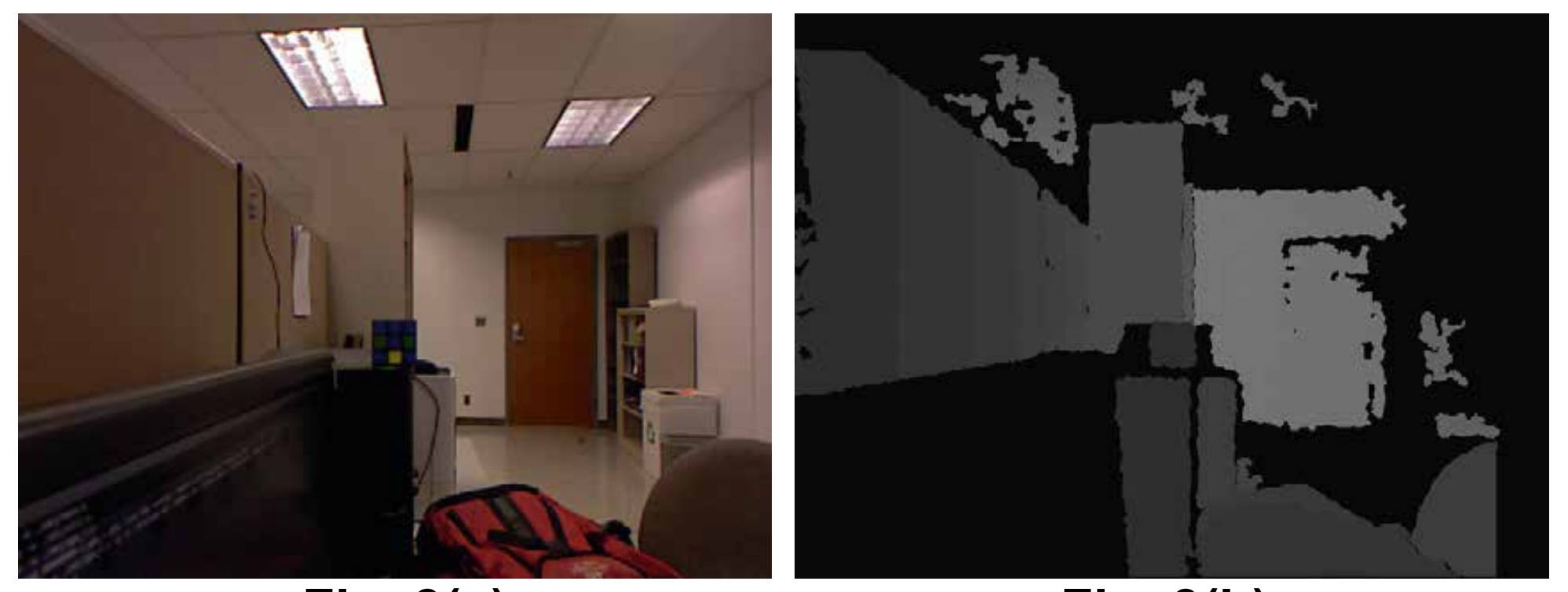
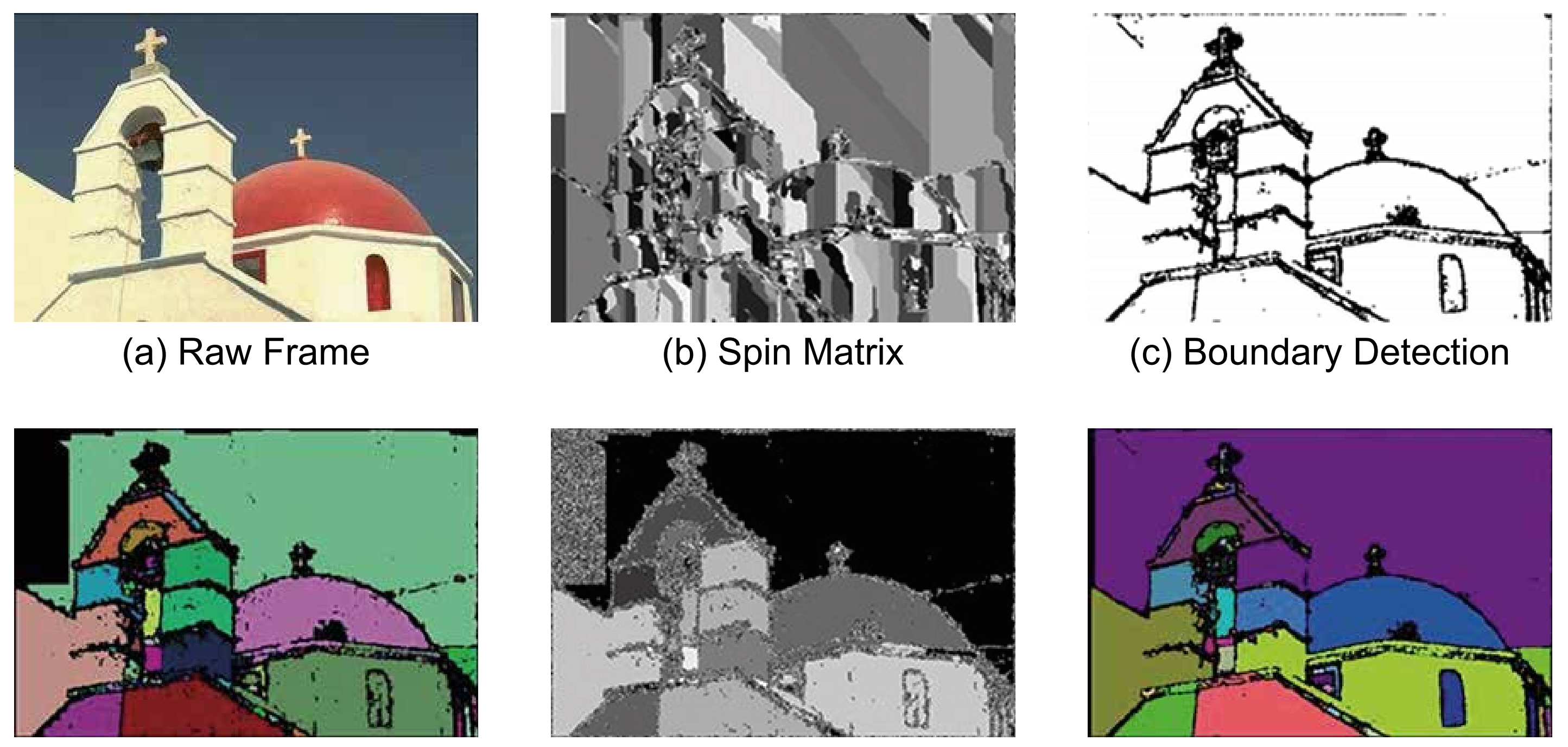


Fig. 3(a) **Fig. 3(b)**

• *Fig. 3* shows the output from an RGB camera *(a)* and an IR camera *(b)*. The depth values in the IR image have been converted to grayscale values. Larger the distance from the IR sensor, higher (whiter) the grayscale output.

4. Segmentation flow


(a) Raw Frame (b) Spin Matrix (c) Boundary Detection
(d) Intermediate Segmentation (e) Reinitialized Spin Matrix (f) Final Segmentation
Fig. 4

5. Optimization strategies

Execution Times (seconds)

Task	***Serial****	***CPU-GPU***
Metropolis Core	3.199	0.013
Auxiliary Tasks	0.006	0.006
Refinement	0.035	0.035
Total	3.230	0.054
Speed-up	---	60x

***The serial implementation was realized on a Xeon x5675 CPU*

• 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.

• 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.

5a. Reduction in Metropolis iterations

• To improve throughput, the number of computations per frame can be reduced by decreasing the number of Metropolis iterations per frame.

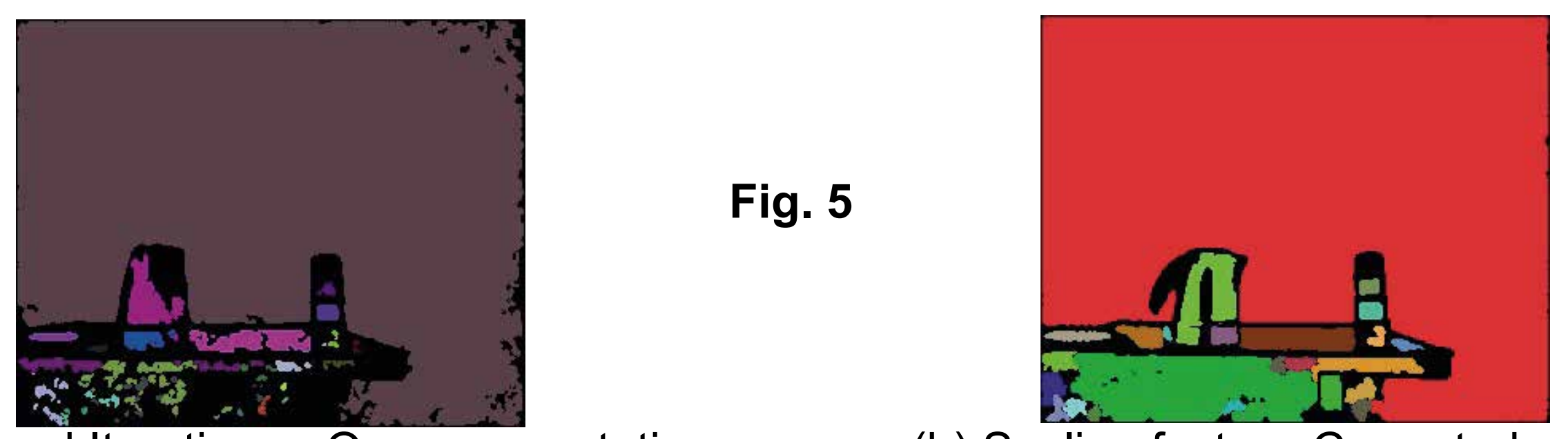


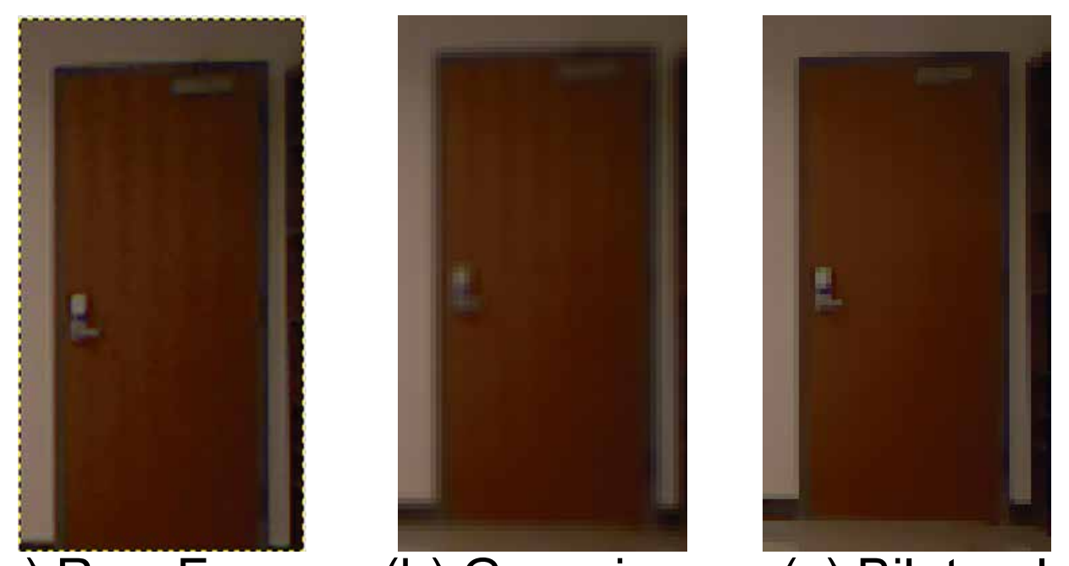
Fig. 5
(a) Reduced Iterations - Oversegmentation (b) Scaling factor - Corrected output

• This, however, leads to an oversegmented frame *(Fig. 5(a))*. Over-segmentation can be interpreted as an outcome of weak interaction strengths between pairs of pixels. Introducing a scaling factor to increase the strengths helps to reduce over-segmentation. The frame rate achieved using this technique is ***44 fps with a slight deterioration in segmentation quality as shown in Fig. 5(b).***

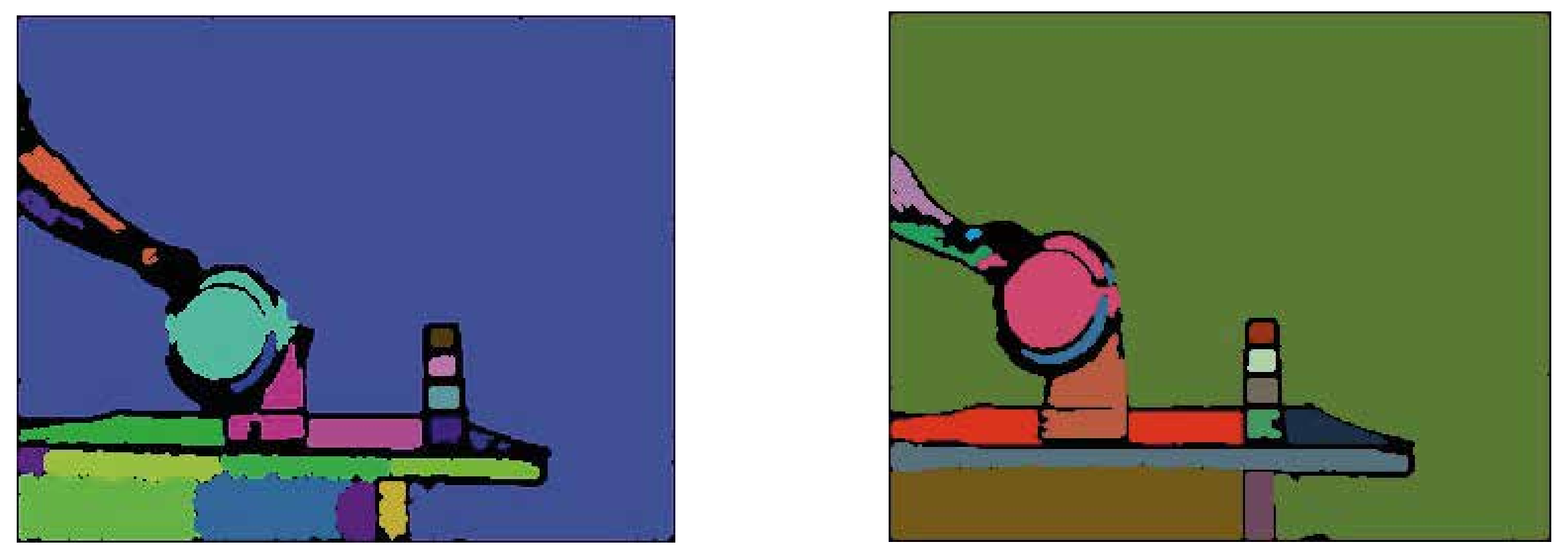
5b. Bilateral filter for noise removal

• 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.

• A bilateral filter is able to get rid of the vertical dark striations from the door which would otherwise cause poor segmentation.


(a) Raw Frame (b) Gaussian (c) Bilateral

• 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).***


(c) Gaussian filtered frame **Fig. 6** (d) Bilateral filtered frame

5c. Depth enhanced coupling matrix

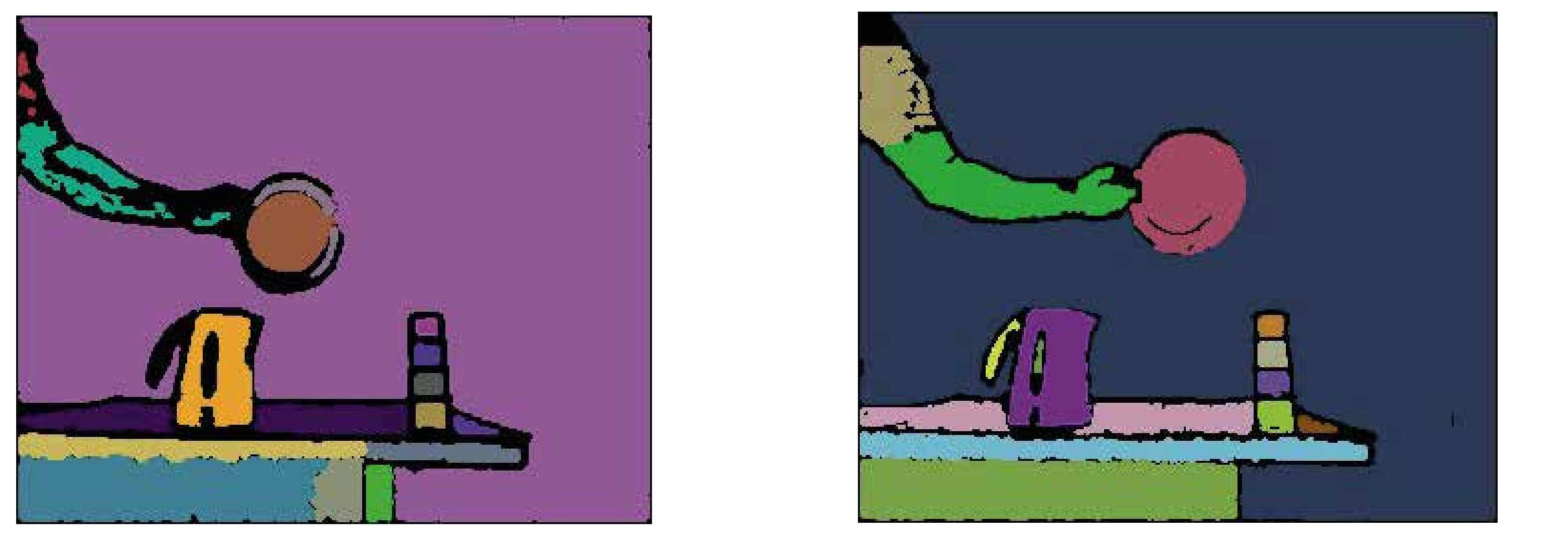
• An alternative to using bilateral filtering would be to use depth measurements in the calculation of coupling matrices. The motivation is to eliminate lighting variation in the 2D planes parallel and perpendicular to the focal plane of the camera.

• For a given color difference (ΔC) and a depth difference (ΔD), we define two thresholds CT and DT respectively. We come up with the following six cases depending on the values of the differences.

Color Difference	**Definition**	***Depth Difference***	**Definition**
Same	$\Delta C = 0$	*Same*	$\Delta D = 0$
Slightly Different	$0 < \Delta C \le CT$	*Slightly Different*	$0 < \Delta D \le DT$
Different	$\Delta C > CT$	*Different*	$\Delta D > DT$

• 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.
• 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].**


(a) Original Flow **Fig. 7** (b) Depth Enhanced

6. References

[1] A. Abramov, K. Pauwels, J. Papon, F. Worgotter, B. Dellen, "Depth- supported real-time video segmentation with the Kinect," wacv, pp.457-464, 2012 IEEE Workshop on the Applications of Computer Vision, 2012
[2] http://www2.engr.arizona.edu/~rcl/huawei/tracking/links.htm