McGill University
School of Computer Science
COMP 558

Project Report: Light Source Estimation using Kinect

Author:
Zhongshi (Sam) Jiang
Shayan Rezvankhah

Supervisor:
Dr. Kaleem Siddiqi

April 26, 2013

Contents

Abstract

In this project, we proposed and evaluated a novel approach to estimate the position of a white point-light source using Kinect color and depth data, assuming Lambertian lighting model. There have been many attempts on light source estimation for 2D images in the past, but to our knowledge, a Kinect system has never been used. Light source estimation problem is ill-posed by nature. The depth data provided by Kinect can help estimate surface normal and hence makes light source estimation possible. To evaluate our approach, we created a simple test scene, with four different lighting positions. Results show that our approach yields quality estimation.

1 Introduction

The light source estimation problem is to solve for the positions of light sources, given the image of a scene. In general, this is a very hard problem because the perceived image is affected by many conjoint factors, such as lighting, surfaces, textures, material, etc. In this project, a single white point-light is assumed to simplify the problem, using the Lambertian lighting model. This simpler problem is related to the shape from shading (SFS) problem presented in class. In SFS, the the lighting and shading information is known, and the goal is to solve for the shape of objects in a scene. In comparison, in this light source estimation problem, the lighting information is to be solved, and the shape and shading information need to be known. To obtain the shape information, Microsoft Kinect is used. It is a commodity sensor device that provides quality depth data of a scene, hence allowing the estimation of shapes.

We proposed a novel approach to tackle this problem, and provided empirical evaluations showing that our approach yields quality light estimations, when using good surfaces.

2 Related Work

There are some previous literature on this topic. They often require strong assumptions.

Light Source Position and Reflectance Estimation from a Single View without the Distant Illumination Assumption[2], by K. Hara. This paper assumes that the 3D model of the objects in a scene is given as input.

Light source estimation of outdoor scenes for mixed reality[3], by Liu, Yanli et al. The paper includes an offline stage to learn the lighting conditions from an image database of the scene and assumes outdoor conditions.

A Practical Method for Estimation of Point Light-Sources[4], by Martin Weber and Roberto Cipolla. This paper assumes a known object geometry.

Automatic Estimation of the Projected Light Source Direction[5], by Peter Nillius and Jan-olof Eklundh. In this paper, only direction is estimated, and it requires occluding objects.

Estimation of the Light Source Distribution and its Use in Integrated Shape Recovery from Stereo and Shading[6], by D. Hougen et al. This paper uses a lighting model with point lights at infinity.

3 Kinect

Kinect is a motion sensing input device by Microsoft, originally designed for the Xbox 360 video games. It has three main components: IR emitter, IR sensor, and RGB camera. The IR emitter projects structured, equal-density grid points in the inferred range, which is invisible to human eyes. The IR sensor detects reflected IR light, and the distortion patterns correspond to the depth information. The RGB camera captures 2D color data of the same scene.

The quality of depth data is rather impressive, given its low cost of roughly $120 Canadian dollars. This commoditized depth sensing device has enabled many interesting projects, such as the Efficient Model-Based 3D Tracking of Hand Articulations using Kinect[1] project, by Forth.

The Kinect IR sensor has a specific range in which the performance is optimal, as shown in Figure 1. In this project, the Default Range model of Kinect is used, which performs optimally against objects at 0.8 meters to 4 meters away. The objects in our created test scene are placed around 0.8 meters away from Kinect, to make them appear as large as possible on screen, while maintaining optimal depth quality. The test scene is described in more details in the Evaluation and Results section.

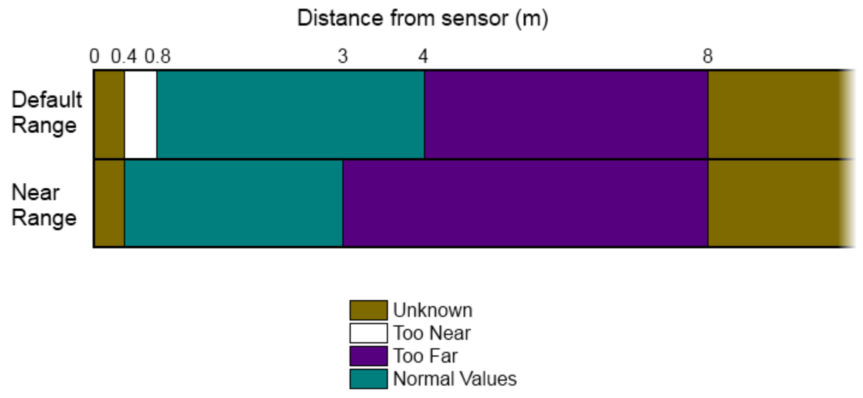

Figure 1: Kinect IR Range

One other characteristic of Kinect is that depth data near object edges is prone to noise, as shown in Figure 2. Algorithm to combat this is planned, but deemed necessary based on preliminary testing results. More on this is explained in later sections.

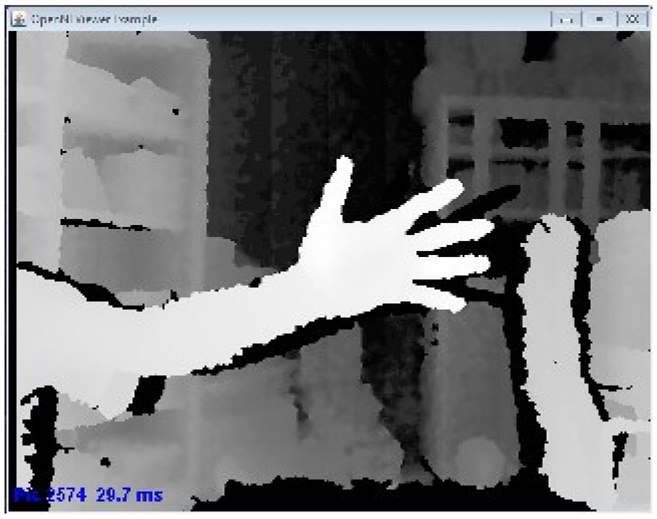

Figure 2: Depth noise near edges. Source: http://fivedots.coe.psu.ac.th/ ad/kinect/ch021/index.html

4 Formulation

This section presents the intuition behind this light source estimation problem, and shows our formulation to this problem.

Recall the reflectance map equation presented in class:

$$R = \rho \vec{i}^T \vec{n}$$

where R is the reflectance map (received light on the RGB camera). ρ is the albedo, property of the materials. $\vec{i}$ is the light source vector, and $\vec{n}$ is the surface normal.

For light sources not at infinite, $\vec{i}$ is changes depending on the location of objects in the scene. Rather than assuming light at infinite, the equation can be re-expressed as:

$$R = \rho(\vec{L} - \vec{P})^T \vec{n}$$

where $\vec{L}$ is the position of the point light, and $\vec{P}$ is the position of a point on an object in the scene. $\vec{L}$ is now fixed for the entire scene. The above equation can be rearranged into the following form:

$$\rho \vec{L} \vec{n} = R + \rho \vec{P}^T \vec{n}$$

Take transposes on both sides:

$$\rho \vec{n}^T \vec{L} = R + \rho \vec{n}^T \vec{P}$$

It's now in the form of $Ax = b$, solvable using Singular Value Decomposition. Each row of A is the normal of a point, and each b_i is the value of $R + \rho \vec{n}_i^T \vec{P}_i$.

The above formulation assumes the albedo ρ is constant for the entire scene. This is a bad assumption because any reasonable scene contain many different materials. Different materials have different albedos, hence contradicting this assumption. Our approach to this problem is to segment the scene, so that the points in the same segment share a constant local albedo. Then the light source position is estimated independently per segment. This is more formally presented in The Algorithm section.

5 The Algorithm

This section describes the details of our method, with short discussions. At a high level, our algorithm consists of the following stages:

- point cloud construction
- normal estimation
- segmentation
- segment selection
- grayscale conversion
- light source estimation per segment

Below presents the details of each stage.

5.1 Point Cloud Construction

Kinect contains two sensors, an IR sensor and a RGB camera. They capture the depths of a scene and 2D RGB image of the scene simultaneously. To construct the combined Point Cloud, reverse perspective projection is done:

$$X_i = \frac{Z_i}{f} x_i$$

$$Y_i = \frac{Z_i}{f} y_i$$

where X_i, Y_i are the 3D coordinates to be recovered, for point i. Z_i is the depth data from the IR sensor. f is the focal length, and x_i, y_i are the pixel coordinates of the RGB data. All the recovered 3D points together are known as the *Point Cloud*.

Figure 3 shows an example of the constructed point cloud. The top image is the recovered 3D depth data, coloured using a colormap. The middle image is the constructed point cloud, where the color of pixels is from the RGB data. The bottom image is the same constructed point cloud, from a bird view. The large, black regions correspond to occluded areas where Kinect sesnor cannot see.

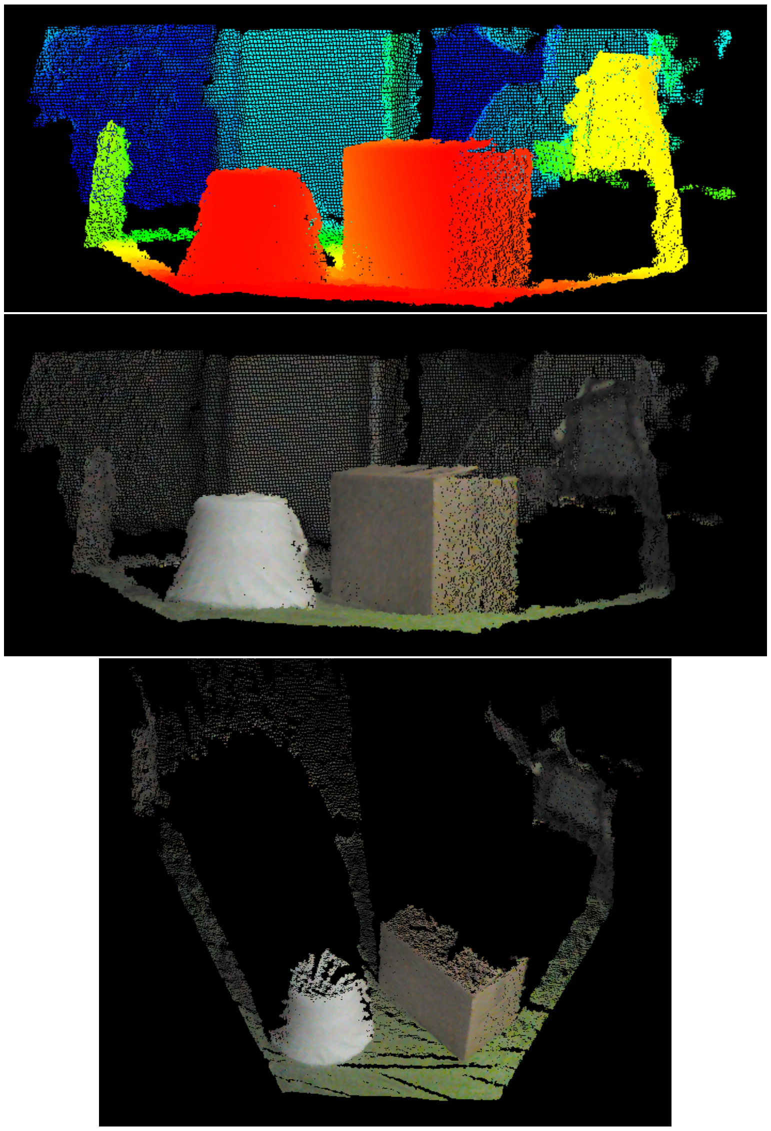

Figure 3: Example of constructed point cloud, showing depth data (top), and the front view (middle) and bird view (bottom).

5.2 Normal Estimation

A least-square plane fitting algorithm is used to estimate the surface normals at each point in the scene. At a high level, this algorithm attempts to estimate a plane, tangent to the surface at a given point, which in turns gives the normal of the surface. This is a least-square plane fitting estimation problem. [7]. One way to solve it is to use PCA on a covariance matrix C, created from the nearest neighbours of the target point:

$$C = \frac{1}{k}\sum_{i=1}^{k}(\vec{p}_i - \bar{p})(\vec{p}_i - \bar{p})^T$$

where k is the number of neighbours to be considered, p_i is the point of interest, and $\bar{p}$ is the centroid in the neighbourhood.

In general, the normals obtained using this PCA approach are ambiguous, because the sign of the normal is unknown. To work around this, the location of Kinect (the viewport) must be known. For simplicity, it is fixed to be the origin of the coordinates system. Then the correct normal can be determined by checking all combinations and look for one that satisfies:

$$\vec{n}_i \cdot (\vec{0} - p_i) > 0$$

That is, the normals must be pointing to the Kinect device. In other words, every surface observable by the Kinect sensor faces Kinect at some angle. (Assuming no semi transparent objects).

More mathematical details on this algorithm are presented in this PhD thesis by Radu Bogdan Rusu[7].

There is one caveat. The selection of k, the number of neighbours, needs to be decided. This parameter depends on the complexity of objects present in a scene. In this project, a trial and error approach is used. Using neighbours within a radius of $3cm$ yield good results for our scene.

Figure 4 shows an example of the estimated normals, shown as little yellow vectors. The normals are quite accurate, on the flat surfaces of the box, the curved surface of the trapezoidal cone, and even on the every edge of the box. The background is just a little further than 4 meters, and hence out of the optimal range of Kinect. Despite this, the normals calculated are still correct.

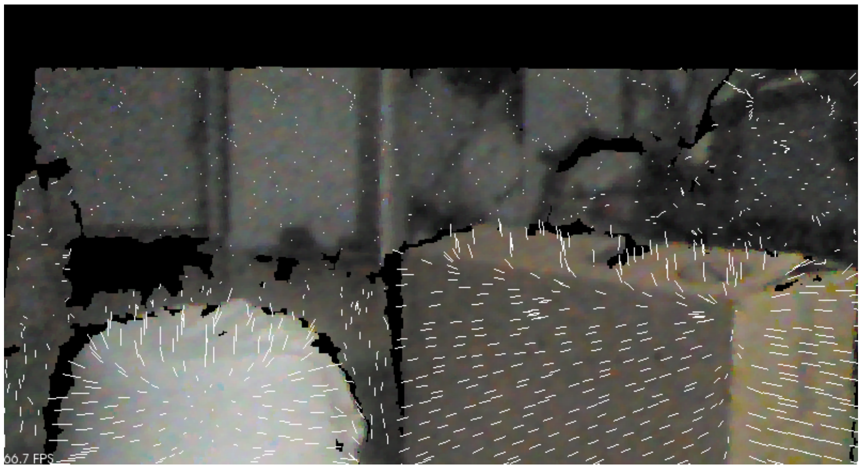

Figure 4: Estimated normals shown as yellow bars

5.3 Segmentation

The goal segmentation is to group points from the same material / object into the same cluster, so that constant local albedo can be assumed within a cluster. The region growing segmentation algorithm[8] is used for this purpose. At a high level, the algorithm works as follows:

```
points <- all points, sorted by curvature
```

```
while points not empty
{
  p <- pop point from points with least curvature
  assign p its own segment S

  do
  {
    for each neighbour point n of S
    {
      if close(n, S) and small_curvature(n)  and similar_color(n, S)
        add n to segment S
    }
  }
  while S is not stable
}
```

The algorithm starts at points with lowest curvature to minimize the number of segments generated. For each starting point, the neighbours are checked and used to grow the segment. `close`, `small_curvature`, and `similar_color` are parametrized functions that test whether a point of interest is close enough to a segment, whether has a small enough curvature, and whether it is of similar colour of a segment. The curvature constraint is loose but important. Sharp edges on a surface can impact the estimation of light source. For the purpose of this project, the parameters are selected manually by trial and error. These parameters are dependent on the lighting condition, materials in the scene and other factors. Automatic selection of these parameters is complex and hence is out of the scope of this project.

Figure 5 shows an example of the segmentation result. Each segment is coloured differently. Overall the segmentation result is fairly nice. Most segments are large, and they correspond roughly to different objects / surfaces of objects.

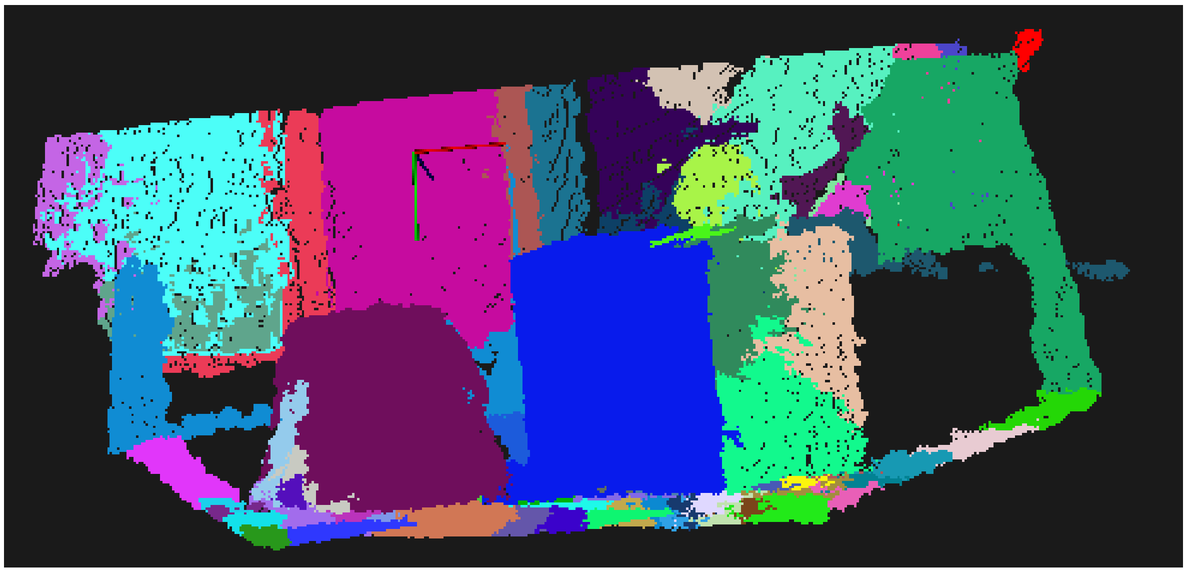

Figure 5: Segmentation result. Different coloring indicate different segments.

5.4 Segment Selection

The segmentation algorithm usually outputs 20-40 segments. In this project, a manual selection is required to indicate which ones should be used for the later light source estimation stage. A UI is developed to aid this process. Each segment is shown in sequential and the user chooses by tapping key “y” or “n”. For each test, two segments are selected: one for the trapezoidal cone and one of the rectangular box.

Figure 6 shows an example of the trapezoidal cone segment selected during this process.

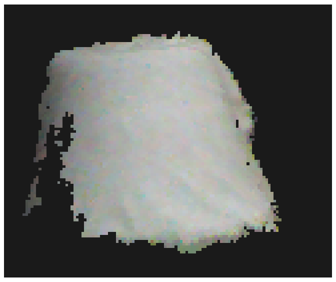

Figure 6: Example of the trapezoidal cone segment presented during the manual selection process.

5.5 Grayscale Conversion

Before estimating the light source, the coloured RGB data is converted to grayscale, using this formula:

$$Y = 0.2126 * red + 0.7152 * green + 0.0722 * blue$$

This version of the formula has default gamma correction embedded.

5.6 Estimating Light Source per Segment

For each segment selected, the corresponding points and their derived information can be plugged into our Formulation to estimate light source location. The actual computation is as simple as solving Singular Value Decomposition.

One workaround is used to solve memory problem. Typically there are 10-20 thousand points in a segment. The underlying SVD library can only support approximately 7 thousand points at a time. To work around it, the following sampling algorithm is used:

```
limit = 7000;
k = number_of_points / limit;

for i = 0..k
  sample = randomly subsample 7000 points;
  L(i) = SVD(sample);
end for

final_L = mean(L(0..k));
```

Statistically, each point is expected to appear in the SVD once, hence contributing to the final estimation. We also tried simply using one subsample, rather than subsampling many times. The result is slightly less stable.

5.7 Notes

This section captures some additional notes about our approach. When starting this project, many potential obstacles were anticipated, and corresponding workarounds were planned. However, after the initial evaluation of the results, these workarounds were deemed unnecessary and never implemented. They are still worth mentioning because they might be necessary in less controlled settings.

5.7.1 Averaging of Temporal Frames

Depth data near object edges may be prone to noise, as mentioned in the Introduction. One workaround is to buffer the data from last few frames, and average them to overcome small noise. Such moving window averaging incurs minimal memory and CPU cost.

5.7.2 Segment Blurring

If objects in scene contain more complex, slightly stronger texture, they may not always be grouped into different segments by the region growing segmentation algorithm. This could start to invalidate the constant local albedo per segment assumption. To combat against it, light segment-local blurring can be done, to smooth out light textures and noise if any.

5.7.3 Different Segmentation Algorithms

One other segmentation algorithm candidate we considered is HSL-based. The intuition is that, a curved surface made of the same material may seem to have different colour under the RGB colorspace. However, in the Hue-Saturation-Lightness[9] (HSL) space, hue and saturation stay relatively constant, even if the lightness changes due to surface normal. Hence HSL-based segmentation may yield better results in the context of light source estimation.

6 Implementation

The project is implemented in C++, on Windows 7, using Visual Studio 2010. A number of libraries are used:

- Point Cloud Library (PCL)[10]: a cutting edge, feature rich, BSD licensed library for image processing. We utilized the following features: point cloud construction, surface normal estimation, and segmentation.
- OpenNI[11]: A framework and library for natural user interface (Kinect) integration.
- Prime Sense Driver[12]: Prime Sense is the company behind the technologies used in Kinect, and they provide Windows drivers for Kinect using the OpenNI framework.
- Eigen[13]: well known, popular, MPLv2 licensed library for linear algebra. We used the SVD feature.
- Many other C++ helper libraries, such as Boost.

The source code can be found at GitHub, LGPLv2.1 licensed.

The recorded scene and corresponding data can be found in this Google Driver Folder.

7 Evaluation and Results

We created a simple scene, and tested it with four different light source locations. Figure 7 shows the configuration of the scene. There are two primary objects. On the left, there is the white trapezoidal cone. On the right, there is the brown rectangular box. Notice the trapezoidal cone has some wrinkles on it, and our results indicate our approach is robust against minor disturbances like such.

Four different light source locations are evaluated: left, top left, right, and top right, relative to the Kinect sensor. The light source is always on the same vertical plane as Kinect, and it's always pointing the scene. The light source used is a household florescent lamp. A hand-made screen is attached to dim and better focus the light. The whole

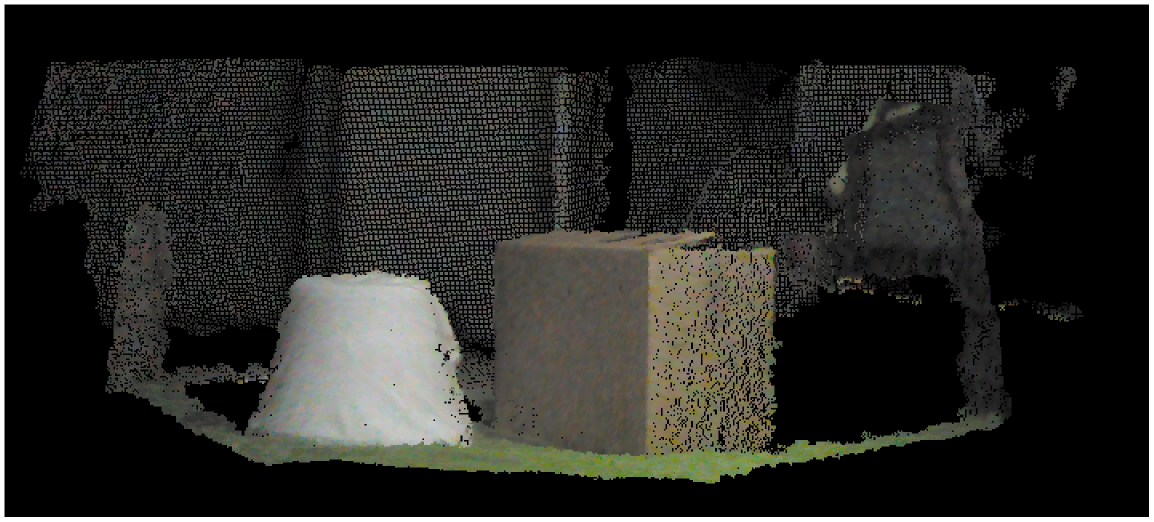

Figure 7: Scene used (screenshot of point cloud)

data gathering process took place in a pseudo darkroom, where the lamp is the only light source, and curtains are closed on windows. The recording is done after sundown, to minimize ambient light. (The room is small and the walls are white. There is still a little bit of light reflecting back.)

Kinect is chosen to be the origin of the world coordinates. A right-handed coordinate system is used, where $+x$ is to the right, $+y$ is down, and $+z$ is forward into the scene.

Table 1 shows the evaluation results.

Test Name	Actual		Est. from Trapezoidal Cone		Est. from Box	
	Position	xy Angle	Position	xy Angle	Position	xy Angle
left	(-0.92, -0.37, 0)	158°	(-0.55, -0.21, -0.81)	159°	(-0.18, -0.18, -0.97)	135°
top left	(-0.71, -0.71, 0)	135°	(-0.45, -0.39, -0.80)	139°	(0.05, -0.55, -0.84)	85°
right	(0.92, -0.37, 0)	22°	(0.42, -0.33, -0.84)	39°	missing	missing
top right	(0.71, -0.71, 0)	45°	(0.45, -0.49, -0.74)	48°	(-0.22, -0.58, -0.79)	111°

Table 1: Evaluation Results. The light source positions are given as unit vector, relative to Kinect. The light source xy angle are given in degrees, relative to +x, counter-clock wise.

7.1 Discussions

Overall, the xy angles estimated from the trapezoidal cone yield high quality estimation of the light source, only off by a few degrees. This result benefits from the nice curvature of the object. On the other hand, xy angles estimated from the box are much worse. This is due to the flat surface of the box. (The measurement from the "right" light location is missing because of a recording failure.) This result indicates that selection of segments is critical in obtaining high quality estimations. The Future Work section presents on the possibility of automatic selection of high quality segments.

On the other hand, the z position of light sources estimated in all scenarios are off significantly. The result indicates the light source being much further behind the Kinect device. The actual z coordinate of the light position is 0. We thought this is due to the influence of weak ambient light, as a result of our imperfect dark room and/or light source. The presence of ambient light simulates a point light source further away from the scene. We attempted to correct this by subtracting the mean segment brightness times a small coefficient from each pixel, to cancel the effect of ambient light. We experimented with the following coefficients: 0.05, 0.1, 0.2, 0.4, but they had no effect on the

z coordinate of the estimated light source.

7.2 Computation Time

The computation is expensive because of the selection of algorithms and the choice of operating at full Kinect resolution, which is 640x480 = 307200 points. The required computation time is measured on a laptop with an Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz and 6GB of DDR3 RAM @ 1600MHz, running Windows 7 64 bit. Table 2 shows the coarse computation time.

Stage	Time in seconds
Kinect driver / device initialization	5-10
Scene capture	$\approx$ 0.05
Normal estimation	30-45
Segmentation	45-60
SVD per 7000 points	10-15

Table 2: Computation time

Overall, our algorithm is expensive. However, improving it to be a real-time algorithm is very possible. This is further discussed in the Future Work section below.

8 Future Work

8.1 Automatic Segment Selection

As shown previously in the Results section, the selection of quality segments in a scene is critical in achieving good performance. Intuitively, a good segment should have certain characteristics. For example, it must be within the optimal range of Kinect. It should be of a certain minimum size. This can be done by estimating the surface area of a segment. A segment should also contain a nice curvature, which is information derivable from the estimated normals. One approach in selecting good segments is to combine these measurable characteristics and possibly more, using a scoring system, where the weights on each metric can be obtained via Machine Learning.

8.2 White Light

The white light assumption can be lifted by removing the color components perpendicular to the light color which would be a generalization of the grayscale conversion. The light color would have to be estimated prior to running the method described in this project and the ambient lighting cancellation may need to be altered.

In this project, one segmentation criteria is the color, and we performed separate light source estimation on each segment independently. A more robust approach would be to consider all the segments' light source position estimations when solving for each segment.

8.3 Improving Computation Time

It's possible to make this light source estimation algorithm real-time, despite the current expensive computation time. The key is to reduce the input size. If scene object locations are assumed to be fixed (or can be estimated during start-up), one can easily filter on the z coordinate of 3D points. This can potentially reduce the input size by orders of magnitudes. Subsequent algorithms would then speed up significantly.

Alternative algorithms can be used in different stages to further improve performance. For example, a faster surface normal estimation algorithm is the greedy triangulation algorithm, presented by Zoltan Csaba Marton et al[14]. This not only speeds on the normal estimation stage, it would reduce the input size to SVD by changing the inputs from points to triangles.

9 Conclusion

In this project, we presented a novel approach to incorporate Kinect into the light source estimation problem. The quality depth data provided Kinect allows shape information to be derived for a scene, and then in turn makes estimating the light source possible. Our evaluations show that this approach performs well when using nicely curved objects / surfaces in the scene.

There are many improvements possible to this project, such as making the algorithm real-time, or lifting some of the assumptions. We are confident that these are very achievable goals, given more time and effort. We have been very excited for this project, and are interested in continuous improvements outside the scope of the course.

10 Assessment of Work Distribution

Much of the work is done in frequent group meetings, so a great part of the project is the result of collective intelligence. Here is an approximate assessment of what each member did:

Shayan Rezvankhah: project idea, theory for basic steps and structure of the method, related work research, background knowledge in vision, drafting of the report.

Zhongshi (Sam) Jiang: formulation, test scene generation and recording, all implementation (Shayan is unable to participate because his laptop is broken) and evaluation, report rewrite, revision and reformat in LaTeX.

References

[1] N. K. I. Oikonomidis and A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," in *22nd British Machine Vision Conference, BMVC'2011*, 2011.

[2] K. N. K. Hara and K. Ikeuchi, "Light source position and reflectance estimation from a single view without the distant illumination assumption," in *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. Vol. 27, April 2005.

[3] Y. Liu, X. Qin, S. Xu, E. Nakamae, and Q. Peng, "Light source estimation of outdoor scenes for mixed reality," *The Visual Computer*, vol. 25, no. 5-7, pp. 637–646, 2009. [Online]. Available: http://dx.doi.org/10.1007/s00371-009-0342-4

[4] M. Weber and R. Cipolla, "A practical method for estimation of point light-sources," in *BMVC 2001*, vol. 2, 2001, pp. 471–480.

[5] P. Nillius and J. olof Eklundh, "Automatic estimation of the projected light source direction," in *CVPR*, 2001, pp. 1076–1083.

[6] D. Hougen and N. Ahuja, "Estimation of the light source distribution and its use in integrated shape recovery from stereo and shading," in *Computer Vision, 1993. Proceedings., Fourth International Conference on*, 1993, pp. 148–155.

[7] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[8] M. Mancas, B. Gosselin, and B. Macq, "B.: Segmentation using a region-growing thresholding," in *Proceedings of the SPIE 5672 (2005) 388–398*, pp. 12–13.

[9] G. H. Joblove and D. Greenberg, "Color spaces for computer graphics," in *Computer Graphics (SIGGRAPH '78 Proceedings)*, 1978.

[10] Point cloud library. [Online]. Available: http://pointclouds.org/

[11] Openni. [Online]. Available: http://www.openni.org/

[12] Prime sense. [Online]. Available: http://www.primesense.com/

[13] Eigen library. [Online]. Available: http://eigen.tuxfamily.org/index.php?title=Main_Page

[14] Z. C. Marton, R. B. Rusu, and M. Beetz, "On Fast Surface Reconstruction Methods for Large and Noisy Datasets," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009.