Princeton University

COS429 Computer Vision

Problem Set 2: Reconstructing a Simpler World

In 1966, Seymour Papert wrote a proposal for building a vision system as a summer project [2]. The abstract of the proposal starts stating a simple goal: "The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system". The report then continues dividing all the tasks (most of which also are common parts of modern computer vision approaches) among a group of MIT students. This project was a reflection of the optimism existing on the early days of computer vision. However, the task proved to be harder than anybody expected.

The goal of this assignment is to embrace the optimism of the 60's and to build an end-to-end visual system. During this process, we will cover some of the main concepts that will be developed in the rest of the course.

Problem 1 *Making the world simpler*

As the visual world is too complex, we will start by simplifying it enough that we will be able to build a simple visual system right away. This was the strategy used by some of the first scene interpretation systems. L. G. Roberts [3] introduced the Block World, a world composed of simple 3D geometrical figures.

For the purposes of this assignment, let's think of a world composed by a very simple (yet varied) set of objects. These simple objects are composed of flat surfaces which can be horizontal or vertical. These objects will be resting on a white horizontal ground plane. We can build these objects by cutting, folding and gluing together some pieces of colored paper as shown in Figure 1. Here, we will not assume that we know the exact geometry of these objects in advance.

Task: Create your own simple world (print Figure 1, mate paper recommend). And take a picture of the world you created and put it in the report.

Problem 2 *Taking orthographic pictures*

One of the simplest forms of projection is parallel (or orthographic) projection. In this image formation model, the light rays travel parallel to each other and perpendicular to the camera plane. This type of projection produces images in which objects do not change size as they move closer or farther from the camera, parallel lines in 3D remain appear as parallel lines in the 2D image. This is different from the perspective projection where the image is formed by the convergence of the light rays into a single point (focal point). If we do not take special care, most pictures taken with a camera will be better described by perspective projection (Figure 2(a)).

Please take pictures with different settings of a camera to create pictures with perspective

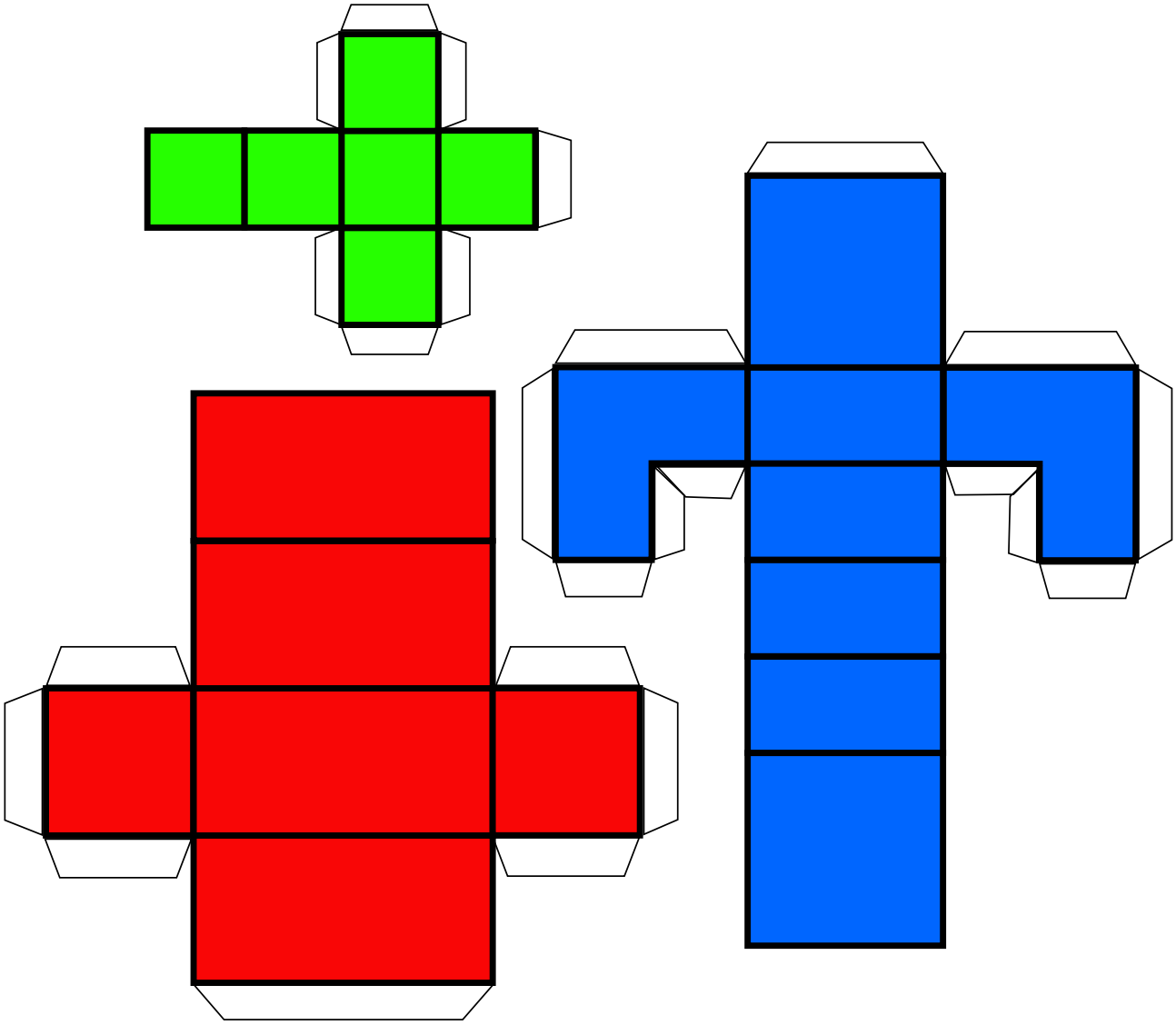

Figure 1: A world of simple objects. Print this page with color to make a simple world.

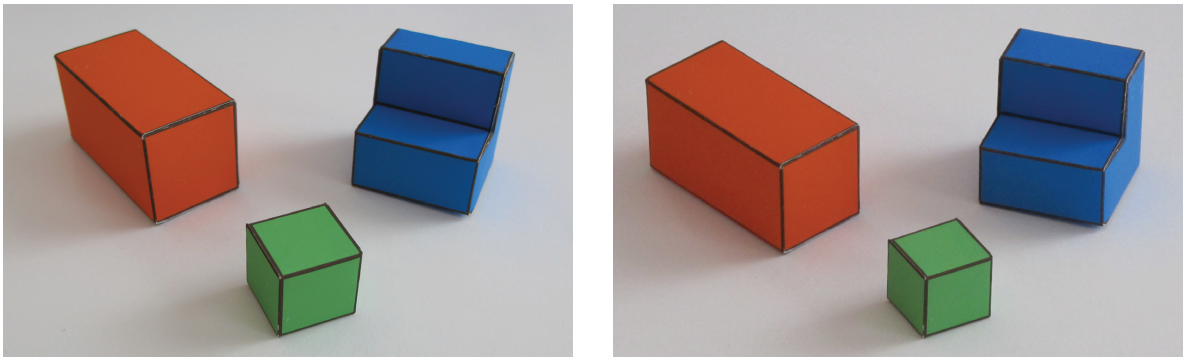

Figure 2: (a) Close up picture without zoom. Note that near edges are larger than far edges, and parallel lines in 3D are not parallel in the image, (b) Picture taken from far away but using zoom. This creates an image that can be approximately described by parallel projection.

projection and with orthographic projection. Both pictures should cover the same piece of the scene.

To create pictures with orthographic projection you can do two things: 1) use the zoom of the camera, 2) crop the central part of a picture. You will have to play with the distance between the camera and the scene, and with the zoom (or amount of cropping) so that both images look as similar as possible only differing in the type of projection (similar to Figure 2).

Task: Submit the two pictures in the report.

Problem 3 *Orthographic projection*

One way of generating images that can be described by parallel projection is to use the camera zoom. If we increase the distance between the camera and the object while zooming, we can keep the same approximate image size of the objects, but with reduced perspective effects (Figure 2(b)). Note how, in Figure 2(b), 3D parallel lines in the world are almost parallel in the image (some weak perspective effects remain).

The first step we need to is to characterize how a point in world coordinates (X, Y, Z) projects into the image plane. Figure 3(a) shows our parameterization of the world and camera. The camera center is inside the plane $X = 0$, and the horizontal axis of the camera (x) is parallel to the ground plane $(Y = 0)$. The camera is tilted so that the line connecting the origin of the world coordinates system and the image center is perpendicular to the image plane. The angle θ is the angle between this line and the Z axis. We will see a more general projection transformation. The image is parametrized by coordinates (x, y). The center of the image is at coordinates (x_0, y_0). The resolution of the image (the number of pixels) will also affect the transformation from world coordinates to image coordinates via a constant factor α (we assume that pixels are square and $\alpha = 1$). Taking into account all these factors, the transformation between world coordinates and image coordinates can be written as:

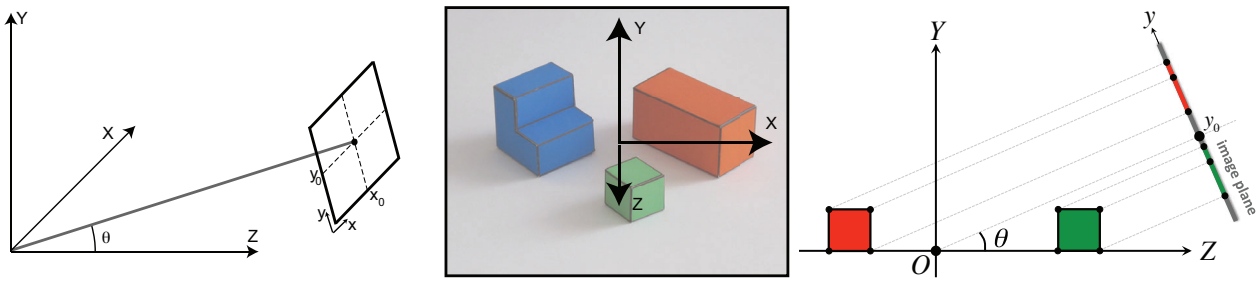

Figure 3: A simple projection model. (a) world axis and camera plane. (b) Visualization of the world axis projected into the camera plane with parallel projection. (c) a 2D profile view for the geometry.

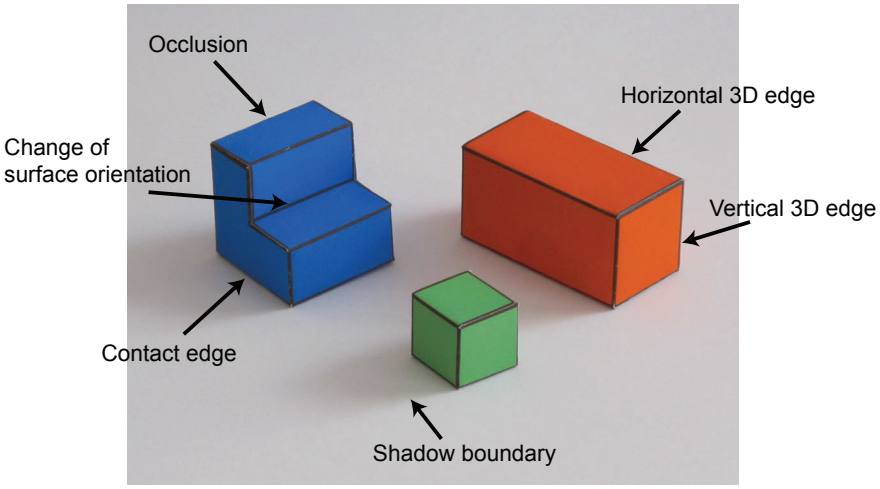

Figure 4: Edges denote image regions where there are sharp changes of the image intensities. Those variations can be due to a multitude of scene factors (occlusion boundaries, changes in surface orienta- tion, changes in surface albedo, shadows, etc.)

$$x = \alpha X + x_0 \quad (1)$$
$$y = \alpha(\cos(\theta)Y - \sin(\theta)Z) + y_0 \quad (2)$$

For this particular parameterization, the world coordinates Y and Z are mixed.

Task: Prove the above two projection equations that relate the coordinates of one point in the 3D world and the image coordinates of the projection of the point in the camera plane.

Problem 4 *Geometric constraints*

Part of the simplification of the vision problem resides in simplifying its goals. In this assignment, we will focus on recovering the world coordinates of all the pixels seen by the camera.

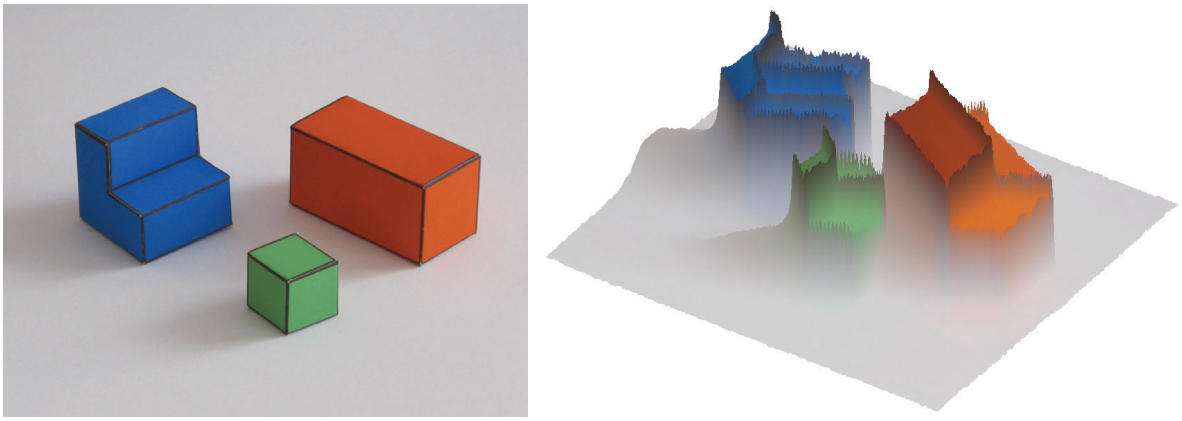

Figure 5: Image as a surface. The vertical axis corresponds to image intensity. For clarity here, I have reversed the vertical axis. Dark values are shown higher than lighter values.

Extracting edges from images Edges denote image regions where there are sharp discontinuities of the image intensities. The first step will consist in detecting candidate edges in the image. Here we will start by making use of some notions from differential geometry. If we think of the image $\mathbf{I}(x, y)$ as a function of two (continuous) variables (Figure 5), we can measure the degree of variation using the gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right) \tag{3}$$

The direction of the gradient indicates the direction in which the variation of intensities is larger. If we are on top of an edge, the direction of larger variation will be in the direction perpendicular to the edge.

However, the image is not a continuous function as we only know the values of the $\mathbf{I}(x, y)$ at discrete locations (pixels). Therefore, we will approximate the partial derivatives by:

$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x - 1, y) \tag{4}$$

$$\frac{\partial \mathbf{I}}{\partial y} \simeq \mathbf{I}(x, y) - \mathbf{I}(x, y - 1) \tag{5}$$

A better behaved approximation of the partial image derivative can be computed by combining the image pixels around (x, y) with the weights:

$$\frac{1}{4} \times \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

From the image gradient, we can extract a number of interesting quantities:

Edge strength:

$$E(x, y) = |\nabla \mathbf{I}(x, y)| \tag{6}$$

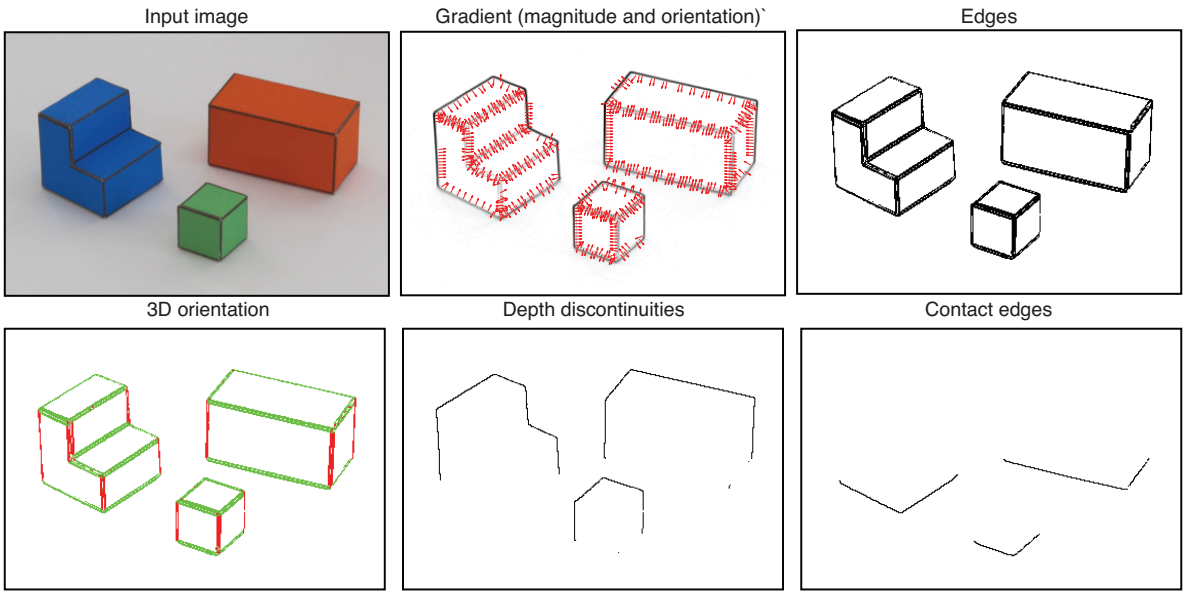

Figure 6: Gradient and edge types.

Edge orientation:

$$\theta(x, y) = \angle \nabla \mathbf{I} = \arctan \frac{\partial \mathbf{I}/\partial y}{\partial \mathbf{I}/\partial x} \tag{7}$$

The unit norm vector perpendicular to an edge is:

$$\mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|} \tag{8}$$

The first decision that we will perform is to decide which pixels correspond to edges (regions of the image with sharp intensity variations). We will do this by simply thresholding the edge strength $E(x, y)$. In the pixels with edges, we can also measure the edge orientation $\theta(x, y)$. Figure 6 visualized the edges and the normal vector on each edge.

From images to surfaces We want to recover world coordinates $X(x, y)$, $Y(x, y)$ and $Z(x, y)$ for each image location (x, y). Given the simple image formation model described before, recovering the X world coordinates is trivial as they are directly observed: for each pixel with image coordinates (x, y), the corresponding world coordinate is $X(x, y) = x$. Recovering Y and Z will be harder as we only observe a mixture of the two world coordinates (one dimension is lost due to the projection from the 3D world into the image plane). Here we have written the world coordinates as functions of image location (x, y) to make explicit that we want to recover the 3D locations of the visible points. In this assignment, we will formulate this problem as a set of linear equations.

Figure/ground segmentation In this assignment, deciding if a pixel belongs to one of the foreground objects or to the background can be decided by simply looking at the color. (However, in general, the problem of segmenting an image into distinct objects is a very

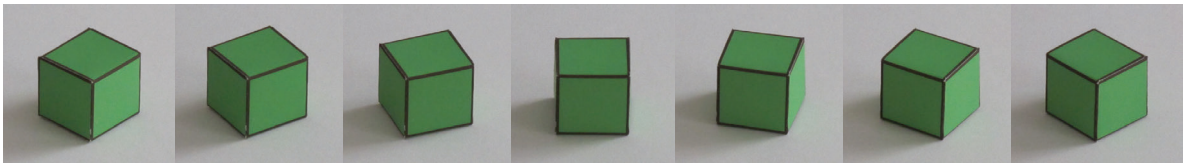

Figure 7: Generic view, accidental alignments.

challenging task.) We just simply set a threshold on the color to tell the foreground and the background apart. If we assume that the background corresponds to an horizontal ground plane, then, for all pixels that belong to **the ground** we can set $Y(x, y) = 0$.

Edge classification In this simple world all edges are either vertical or horizontal. Under parallel projection, we will assume the 2D vertical edges are also 3D vertical edges. Under parallel projection and with the camera having its horizontal axis parallel to the ground, we know that vertical 3D lines will project into vertical 2D lines in the image. On the other hand, horizontal lines will project into oblique lines. Therefore, we can assume than any vertical line in the image is also a vertical line in the world. As shown in Figure 7, in the case of the cube, there is a particular viewpoint that will make an horizontal line project into a vertical line, but this will require an accidental alignment between the cube and the line of sight of the observer. Nevertheless, this is a weak property and accidental alignments such as this one can be common. But it will be good enough for the purposes of this assignment. In Figure 6 we show the edges classified as vertical or horizontal using the edge angle. Anything that deviates from verticality in 15 degrees is labeled as horizontal.

Edge constraint We can now translate the 3D edge orientation into linear constraints. We will formulate these constraints in terms of $Y(x, y)$.

In **a 3D vertical edge**, using the projection equations, the derivative of Y along the edge will be:

$$\partial Y/\partial y \quad = \quad 1/\cos(\theta) \tag{9}$$

In **a 3D horizontal edge**, the coordinate Y will not change. Therefore, the derivative along the edge should be zero:

$$\partial Y/\partial \mathbf{t} \quad = \quad 0 \tag{10}$$

where the vector $\mathbf{t}$ denotes direction tangent to the edge, $\mathbf{t} = (-n_y, n_x)$. We can write this derivative as a function of derivatives along the x and y image coordinates:

$$\partial Y/\partial \mathbf{t} = \nabla Y \cdot \mathbf{t} = -n_y \partial Y/\partial x + n_x \partial Y/\partial y \tag{11}$$

When the edges coincide with occlusion edges, special care should be taken so that these constraints are only applied to the surface that owns the boundary.

Task: In the previous explanation, we have written all the derivative constraints for $Y(x, y)$. The task for this problem is to write the constraints for $Z(x, y)$ in the report PDF file.

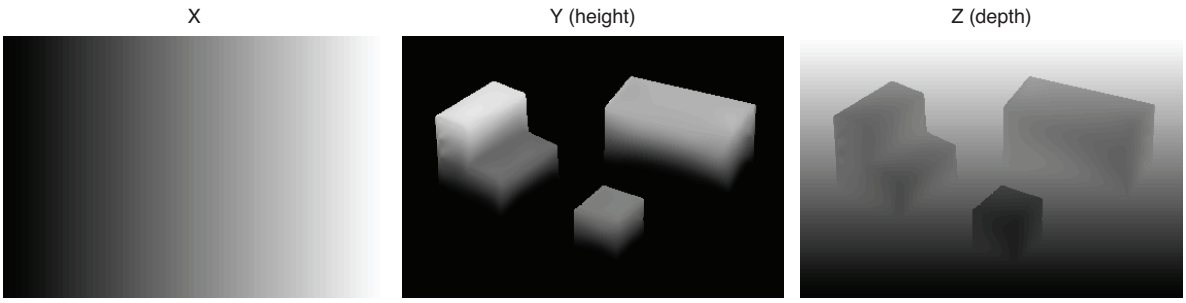

Figure 8: World coordinates corresponding to each image location.

Problem 5 *Approximation of derivatives*

In the points where there are flat image regions (there are no edges) we do not have enough information to decide locally what is the surface orientation. Therefore, we need some criteria in order to propagate information from the boundary (this problem is common in many visual domains). In this case we will assume that the object faces are planar.

$$\partial^2 Y/\partial x^2 = 0 \quad (12)$$
$$\partial^2 Y/\partial y^2 = 0 \quad (13)$$
$$\partial^2 Y/\partial y \partial x = 0 \quad (14)$$

This approximation to the second derivative can be obtained by applying twice the first order derivative approximated by [-1 1]. The result is: $\begin{bmatrix}-1 & 2 & -1\end{bmatrix}$ which corresponds to $\partial^2 Y/\partial x^2 \simeq 2Y(x,y) - Y(x+1,y) - Y(x-1,y)$, and similarly for $\partial^2 Y/\partial y^2$

Task: The task of this problem is to fill the missing kernels (Lines 171 and 187) in the script: ***simpleworldY.m***. You should the rest of the code in this file to figure out how everything works so that you know what to fill in for the missing part. Please copy two lines of code that you wrote (don't include the original code we provided) in your report PDF file.

Problem 6 *A simple inference scheme*

All the different constraints described before can be written as an overdetermined system of linear equations. Each equation will have the form:

$$a_i \mathbf{Y} = \mathbf{b_i} \quad (15)$$

Note that there might be many more equations than there are image pixels.

We can translate all the constrains described in the previous sections into this form. For instance, if the index i corresponds to one of the pixels inside one of the planar faces of a foreground object, then the planarity constraint can be written as $a_i = [0, \ldots, 0, -1, 2, -1, 0, \ldots, 0]$, $b_i = 0$.

We can solve the system of equations by minimizing the next cost function:

$$J = \sum_i (a_i \mathbf{Y} - \mathbf{b_i})^\mathbf{2} \quad (16)$$

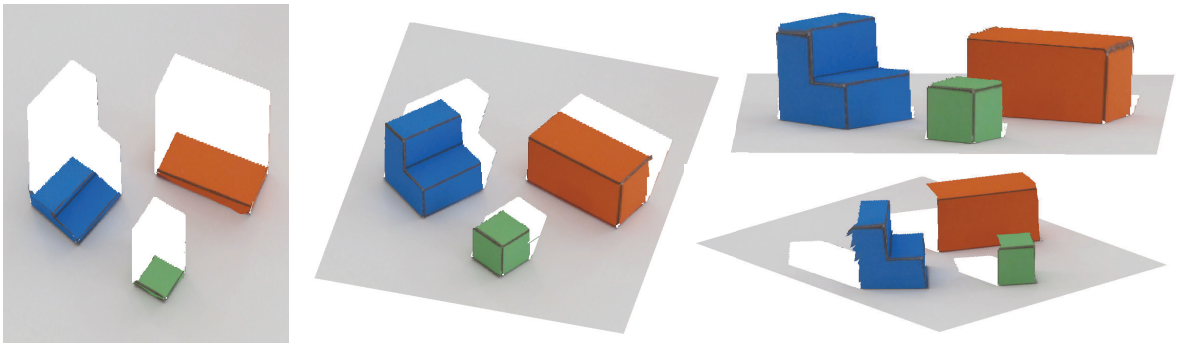

Figure 9: Reconstructed 3D scene structure and synthesis of new viewpoints.

If some constraints are more important than others, it is possible to also add a weight w_i.

$$J = \sum_i w_i (a_i \mathbf{Y} - \mathbf{b_i})^2 \tag{17}$$

It is a big system of linear constraints and it can also be written in matrix form:

$$\mathbf{AY} = \mathbf{b} \tag{18}$$

where row i of the matrix $\mathbf{A}$ contains the constraint coefficients a_i. This problem can be solved efficiently as the matrix A is very sparse (most of the elements are zero).

Figure 8 shows the world coordinates $X(x, y)$, $Y(x, y)$, $Z(x, y)$ for each pixel. The world coordinates are shown here as images with the gray level coding the value of each coordinate (black represents 0). Figure 9 shows the reconstructed 3D scene rendered under different view points.

Task: Run the code with your answers to the previous questions. Select some of the images included with the code and show some new view points for them. Take some screen shots and include in the report PDF file.

Extra Credit [optional] *Violating simple world assumptions*

We can also run the algorithm with shapes that do not satisfy the assumptions that we have made for the simple world. Figure 10 shows the *impossible steps* figure from [1]. Figure 10 shows the reconstructed 3D scene for this unlikely image. The system has tried to approximate the constraints, as for this shape it is not possible to exactly satisfy all the constraints.

Find one of your images when the recovery of 3D information fails. Explain why it fail.

Extra Credit [optional] *The real world*

A research problem is a question for which we do not know the answer. In fact, there might not even be an answer. This question is optional and could be extended into a larger course project.

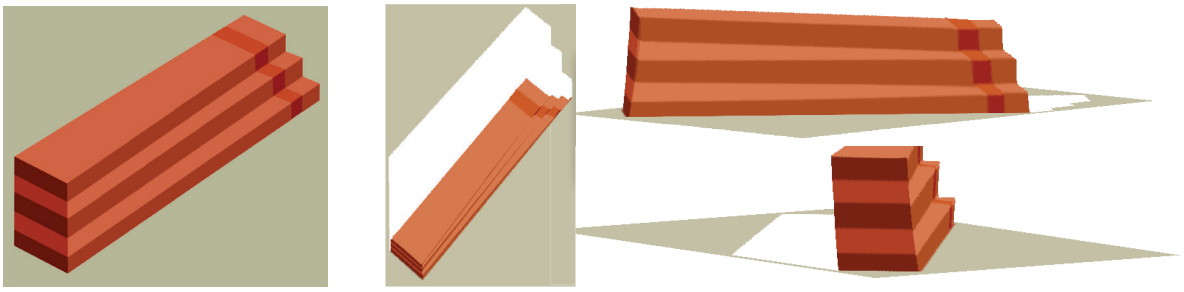

Figure 10: Reconstructed 3D scene structure and synthesis of new viewpoints.

The goal of this problem is to test the 3D reconstruction code with real images. A number of the assumptions we have made will not work when the input are real images of more complex scenes. For instance, the simple strategy of differentiating between foreground and background segmentation will not work with other scenes.

Try taking pictures of real world scenes (not the cubes) and propose modifications to the scheme proposed in this assignment so that you can get some better 3D reconstructions. The goal is not to build a general system, but to be able to handle a few more situations.

What to submit: You need to submit two files: one PDF file for the report that contains your name, Princeton NetID, all the pictures taken and text to answer the questions; one ZIP file (not RAR or any other format) that contains all source code for your system, and a "simpleworldY.m" file that takes no parameter as input and run directly in Matlab to generate the results reported in your PDF file. Both the PDF and ZIP file should be named using your Princeton NetID underscore cos429ps2. As an example using my account, they should be named "xj_cos429ps2.pdf" and "xj_cos429ps2.zip". To **verify your result** and **detect plagiarism** to make sure there is no cheating, we will use an automatic program to run your code and compare your code with other students' (including both this year and all previous years) and public available implementations (e.g. from Google, Bing, Github). Therefore, please follow the file format to make our grading job easier. Failure to follow these rules will result in losing your grade.

Acknowledgement: This assignment is designed based on a related assignment from MIT6.869.

References

[1] Edward H. Adelson. Lightness perception and lightness illusion. In M. Gazzaniga, editor, *The New Cognitive Neurosciences*, pages 339–351. 2000.

[2] Seymour Papert. The summer vision project. MIT AI Memo 100, Massachusetts Institute of Technology, Project Mac, 1966.

[3] Lawrence G. Roberts. *Machine Perception of Three-Dimensional Solids.* Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York, 1963.