Chapter 1

A Simple Vision System

1.1 Introduction

In 1966, Seymour Papert wrote a proposal for building a vision system as a summer project [4]. 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 first chapter is to present several of the main topics that we will cover during this course. We will do this in the framework of a real, although a bit artificial, vision problem.

1.2 Why is vision hard?

It is difficult to say exactly what makes vision hard as we do not have a solution yet [3]. In this section we will mention two of the aspects that make vision hard: first is the structure of the input, and second is the structure of the desired output.

1.2.1 The structure of ambient light

From a light source a dense array of light rays emerges in all directions. But before these light rays reach the eye, the light interacts with the objects in the world. Let's consider a single ray emerging from the light source (we will use here a simple geometric interpretation of the structure of light). If the light ray is not directed towards the eye, this ray will strike some surface in the world and, as result, a new set of rays will be emitted in many directions. This process will continue producing more and more interactions and filling the space. In the middle of the space, the observer will sense only a subset of the light rays that will strike the eye. As a result of this complex pattern of interactions (reflections, absorptions, etc.) even a single ray will form a complex image in the eye of the observer. Figure 1.1 shows some pictures taken when illuminating a complex scene with a laser pointer which will illuminate the scene with a narrow beam of light (this is the best approximation to the image produced by a single light ray that I could do at home). The resulting images show the complexity of the interactions between the different surfaces. The sum of the contribution of all the light rays emitted by the light source will give rise to a natural looking picture. Despite the complexity of the structure of the ambient light (a term coined by J. J. Gibson []), with multiple reflections, shadows, specular surfaces (which provide incorrect disparity information to

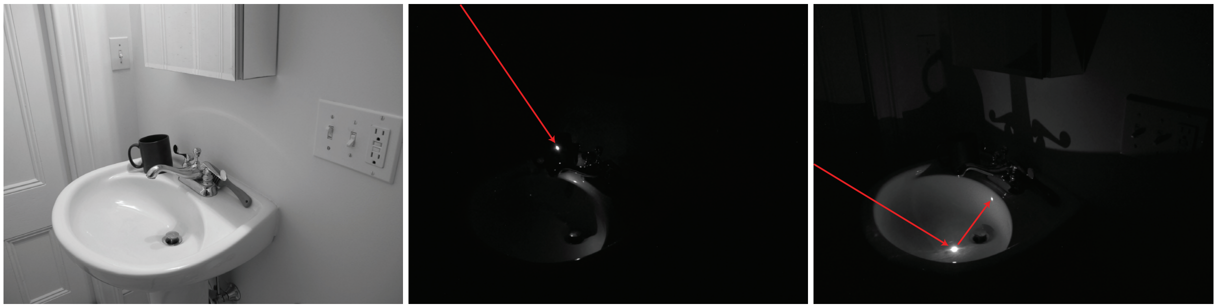

Figure 1.1: Left) scene illuminated with a ceiling lamp. Right) the two Images on the right have been obtained by illuminating the scene with a laser pointer. On each image, it is indicated (red arrow) the approximate direction of the light beam produced by pointer.

our two eyes), our visual system has no problem in interacting with this scene, even if it is among the first things that we have to see just after waking up.

The pattern of light filling the space can be described by the function:

$$P(\theta, \Phi, \lambda, t, X, Y, Z) \tag{1.1}$$

where P is the light intensity of a ray passing by the world location (X, Y, Z) in the direction given by the angle (θ, Φ), and with wavelength λ (we will study color in chapter 7) at an instant in time t. This function, called the plenoptic function by Adelson[1] and Bergen [2], contains all the information needed to describe the complete pattern of light rays that fills the space. The plenoptic function does not include information about the observer. The observer does not has access to the entire plenoptic function, only to a small slice of it. In lecture 11 we will describe how different mechanisms can produce images by sampling the ambient light in different ways.

For a given observer, most of the light rays are occluded. If it was not because of occlusion, vision would be a lot simpler. Occlusion is the best example of how hard vision can get. Many times, properly interpreting an image will require to understand what is occluded (e.g., we know that a person is not floating just because the legs are occluded behind a table). Unfortunately occlusions are common. In fact, in the world, there is more occluded matter than visible to any given observer.

Although knowing the entire plenoptic function can have many applications, fortunately, the goal of vision is not to recover this function.

1.2.2 Measuring light vs. measuring scene properties

If the goal vision was to simply measure the light intensity coming from a particular direction of space (like a photometer) then things will be clear. However, the goal of vision is to provide an interpretation of the world in front of the observer in terms of "meaningful" surfaces, objects, materials, etc., in order to extract all the different elements that compose the scene (anything that will be relevant to the observer). This problem is hard because most of the information is lost and the visual system needs to make a number of assumptions about the structure of the visual world in order to be able to recover the desired information from a small sample of the plenoptic function. It is also hard because of our understanding about what is relevant for the observer is also incomplete.

[1]Adelson was going to call it the holoscopic function, but a well-known holographer told him that he would punch him in the nose if he called it that

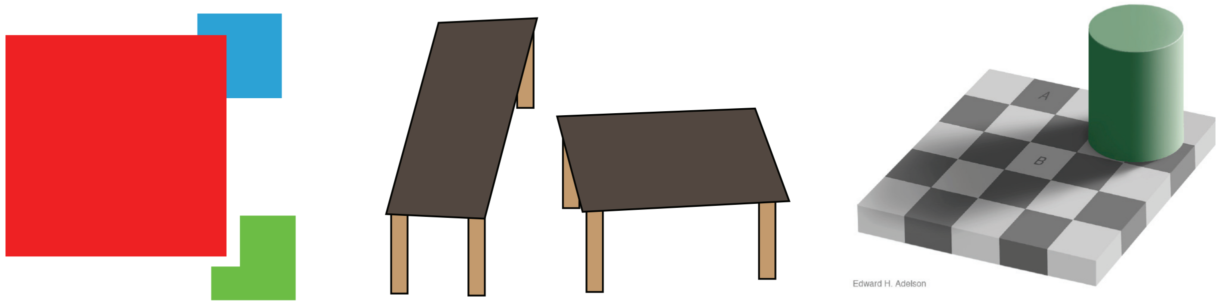

Figure 1.2: The goal is not to measure light intensities, but to extract scene properties relevant for the observer. Even if we are aware of the way this images have been created, we can not shut down the automatic mechanisms that try to interpret this images not as light patterns but as pictures of real 3D scenes.

Figure 1.2 shows different images that illustrate that the human visual system is trying to recover the scenes that are the cause of those images. In the first image, we see a red square occluding a smaller blue square. This is the most natural interpretation of the scene if we assume that squares are typical in the world and that occlusions are common. Even though the green figure shows us that L-shaped figures are possible, we interpret the blue L-shaped figure as an occluded square. The next image shows the ”turning tables” illusion by Roger Shepard. In this illusion, both table tops have the same shape and size but one is rotated with respect to the other. The visual system insists in interpreting this objects as 3D objects giving the impression that the left table is longer than the table of the right. And this perception can not be shut down even if we know exactly how this image has been generated. The third example shows the Checkershadow illusion by E. Adelson. In this figure, the squares marked with A and B have the exact same intensity values. But the visual system may be trying to measure the real intensity of the squares that are part of a 3D scene with a shadow. The visual system tries to remove the effect of the shadow to infer the true gray value of each square. As a result, the square in the shadow is perceived as being lighter than the square outside the shadow region, despite that both have the exact same gray levels.

1.3 Making vision simple

The goal of the rest of this lecture 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.

Vision has many different goals (object recognition, scene interpretation, 3d interpretation, etc) but in this chapter we’re just focussing on the task of 3d interpretation.

1.3.1 A simple world: The blocks world

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 [5] introduced the Block World, a world composed of simple 3D geometrical figures.

For the purposes of this lecture, 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

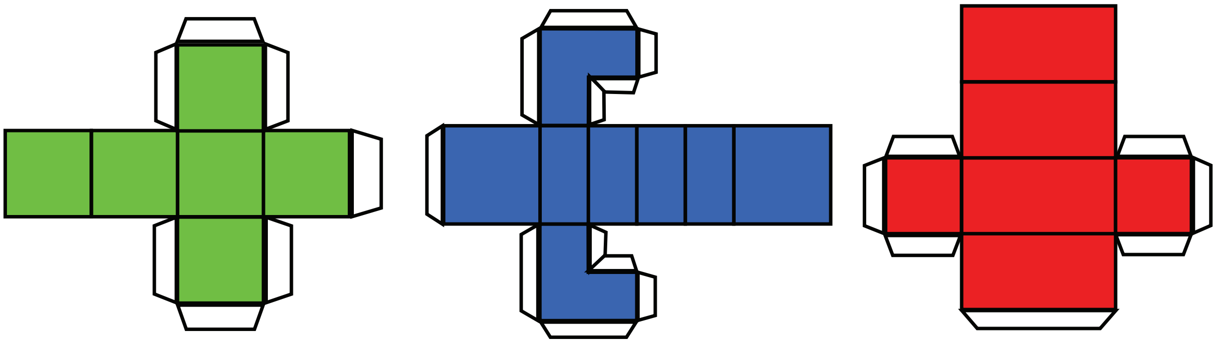

Figure 1.3: A world of simple objects.

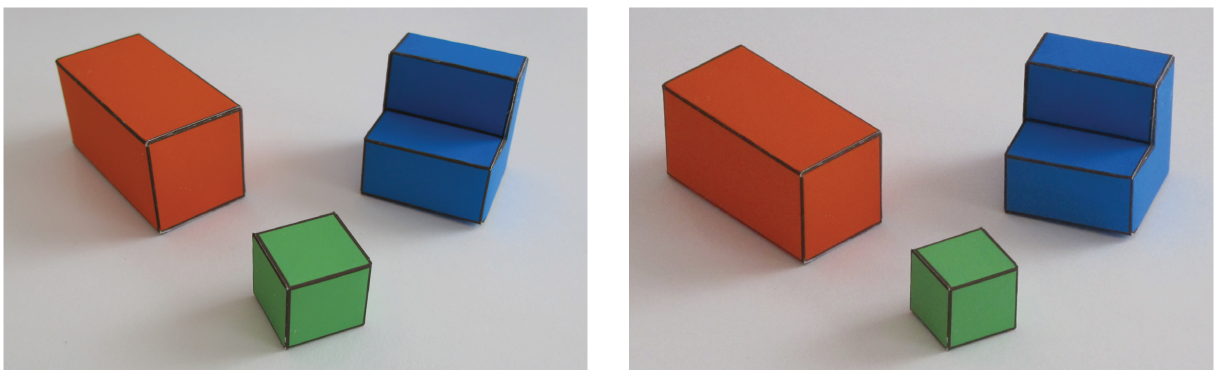

Figure 1.4: a) Close up picture without zoom. Note that near edges are larger than far edges, and parallel lines in 3D are not parallel in the image, b) Picture taken from far away but using zoom. This creates an image that can be approximately described by parallel projection.

and gluing together some pieces of colored paper as shown in figure 1.3. Here, we will not assume that we know the exact geometry of these objects in advance.

1.3.2 A simple image formation model

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 (to be discussed in lectures 11-13) 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 (fig. 1.4.a).

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 (fig. 1.4.b). Note how, in fig. 1.4.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 1.5.a shows our parameterization of the world and camera. The camera center is

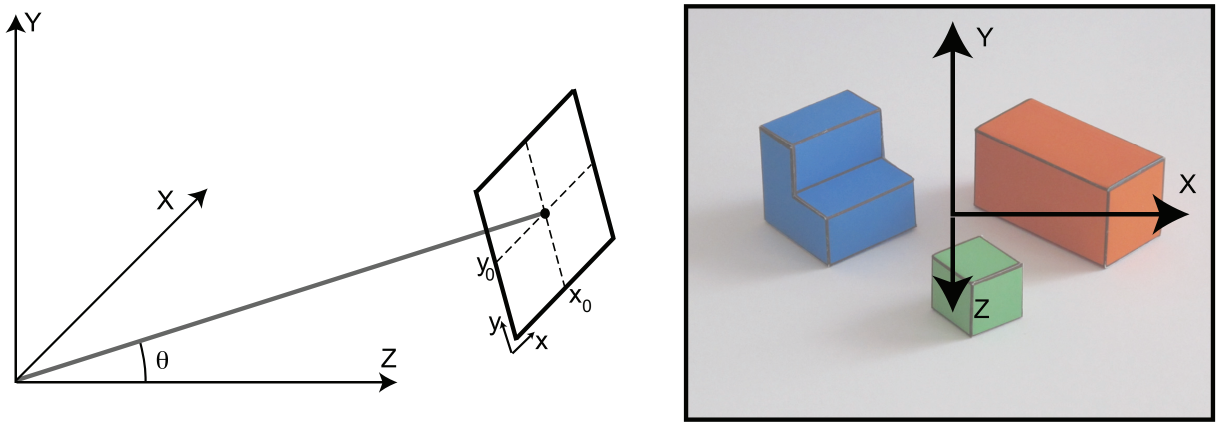

Figure 1.5: A simple projection model. a) world axis and camera plane. b) Visualization of the world axis projected into the camera plane with parallel projection. Note that the Z axis does not look like is pointing towards the observer. Instead, the Z axis is identical to the Y axis up to a sign change and a scaling. A point moving parallel to the Z axis will be indistinguishable from a point moving down parallel to the Y axis.

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 in lectures 11-13. 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 (for now we assume that pixels are square and we will see a more general form in lectures 11-13). Taking into account all these factors, the transformation between world coordinates and image coordinates can be written as:

$$x = \alpha X + x_0 \quad (1.2)$$

$$y = \alpha(\cos(\theta)Y - \sin(\theta)Z) + y_0 \quad (1.3)$$

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

1.3.3 A simple goal

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

Besides recovering the 3D structure of the scene there are many other possible goals that we will not consider in this lecture. For instance, one goal (that might seem simpler but that it is not) is to recover the actual color of the surface seen by each pixel (x, y). This will require discounting for illumination effects as the color of the pixel is a combination of the surface albedo, and illumination (color of the light sources, and inter-reflections).

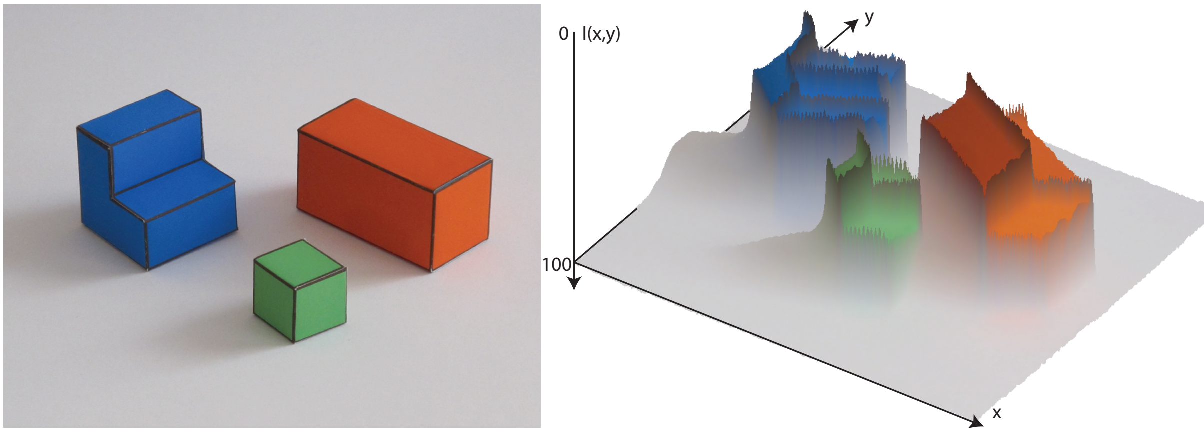

Figure 1.6: Image as a surface. The vertical axis corresponds to image intensity. For clarity here, I have reversed the vertical axis. Dark values are shown higher than lighter values.

1.4 Extracting useful features from the image

The observed image is:

$$I(x, y) \tag{1.4}$$

In this representation, the image is an array of intensity values (color values). This representation is great if we are interested in knowing the light intensity coming from each direction of the space and striking the camera plane as this information is explicitly represented. However, here we are interested in interpreting the 3D structure of the scene and the objects within. Therefore, it will be useful to transform the image into a representation that makes more explicit some of the important parts of the image that carry information about the boundaries between objects, changes in the surface orientation and so on.

The representation will not be unique. Different levels of the scene understanding process will rely on different representations. For instance, the array of pixel intensities $I(x, y)$ is a reasonable representation for the early stages of visual processing as, although we do not know the distance of surfaces in the world, the direction of each light ray in the world is well defined. However, other initial representations could be used by the visual system (e.g., images could be coded in the Fourier domain, or pixels could combine light coming in different directions, etc.). [Question: are there other animal eyes that produce a different initial representation?]

There are several representations that can be used as an initial step for scene interpretation. Images can be represented as collections of small image patches, regions of uniform properties, edges, etc.

1.4.1 A catalog of edges

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

One of the tasks that we will solve first is to classify image edges according to what is the cause (fig. 1.7):

- Object boundaries:
- Changes in surface orientation:

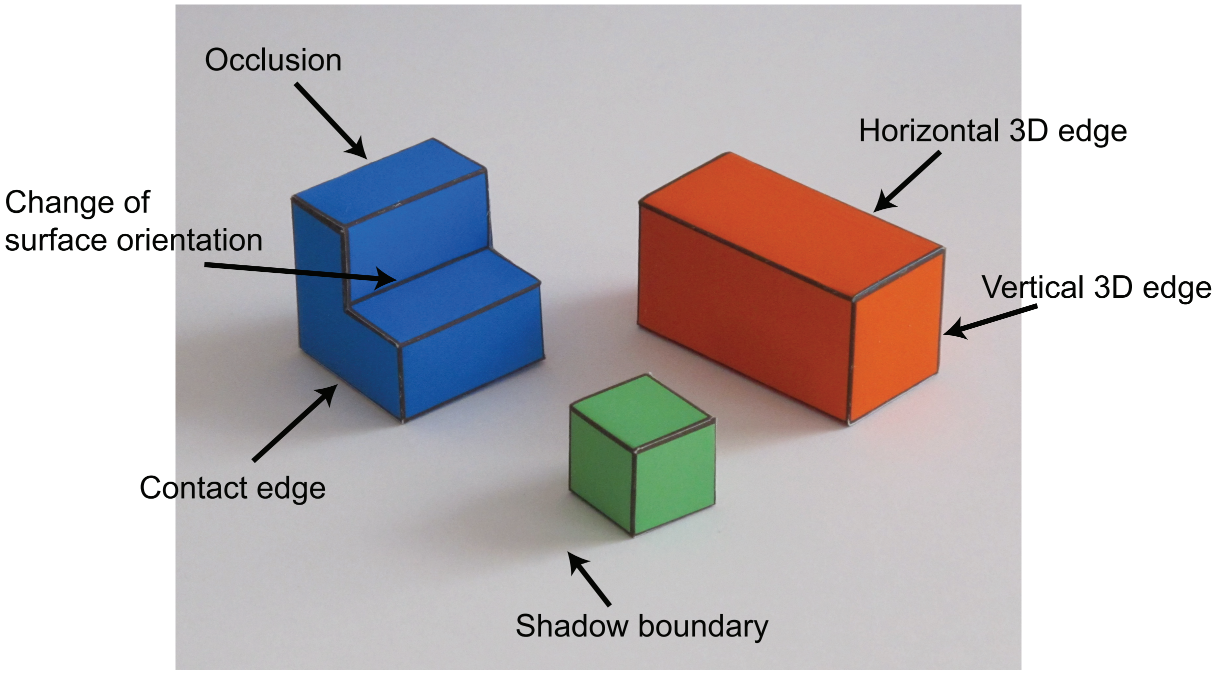

Figure 1.7: Edges denote image regions where there are sharp changes of the image intensities. Those variations can be due to a multitude of scene factors (occlusion boundaries, changes in surface orientation, changes in surface albedo, shadows, etc.)

- Occlusion boundaries: in this simple world, we will position the objects in such a way that objects do not occlude each other.
- Contact edges: this is an edge between two objects but with no depth discontinuity.
- Shadow edges: this can be harder than it seems. In this simple world, shadows are soft creating slow transitions between dark and light.

Despite the apparent simplicity of this task, in most natural scenes, this classification is very hard and requires the interpretation of the scene at different levels. In other chapters we will see how to make better edge classifiers (for instance by propagating information along boundaries, junction analysis, inferring light sources, etc.).

1.4.2 Extracting edges from images

The first step will consist in detecting candidate edges in the image. We will describe other techniques in detail in lecture 14. 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 (fig. 1.6), we can measure the degree of variation using the gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right) \tag{1.5}$$

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:

$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x-1, y) \quad (1.6)$$

$$\frac{\partial \mathbf{I}}{\partial y} \simeq \mathbf{I}(x, y) - \mathbf{I}(x, y-1) \quad (1.7)$$

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

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

We will discuss these approximations in detail in lectures 2-4.

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

Edge strength:

$$E(x, y) = |\nabla \mathbf{I}(x, y)| \quad (1.8)$$

Edge orientation:

$$\theta(x, y) = \angle \nabla \mathbf{I} = \arctan \frac{\partial \mathbf{I} / \partial y}{\partial \mathbf{I} / \partial x} \quad (1.9)$$

The unit norm vector perpendicular to an edge is:

$$\mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|} \quad (1.10)$$

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 lecture 14 we will discuss more sophisticated edge detectors. In the pixels with edges, we can also measure the edge orientation $\theta(x, y)$. Figure 1.8 visualized the edges and the normal vector on each edge.

1.5 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 simple world, we will formulate this problem as a set of linear equations.

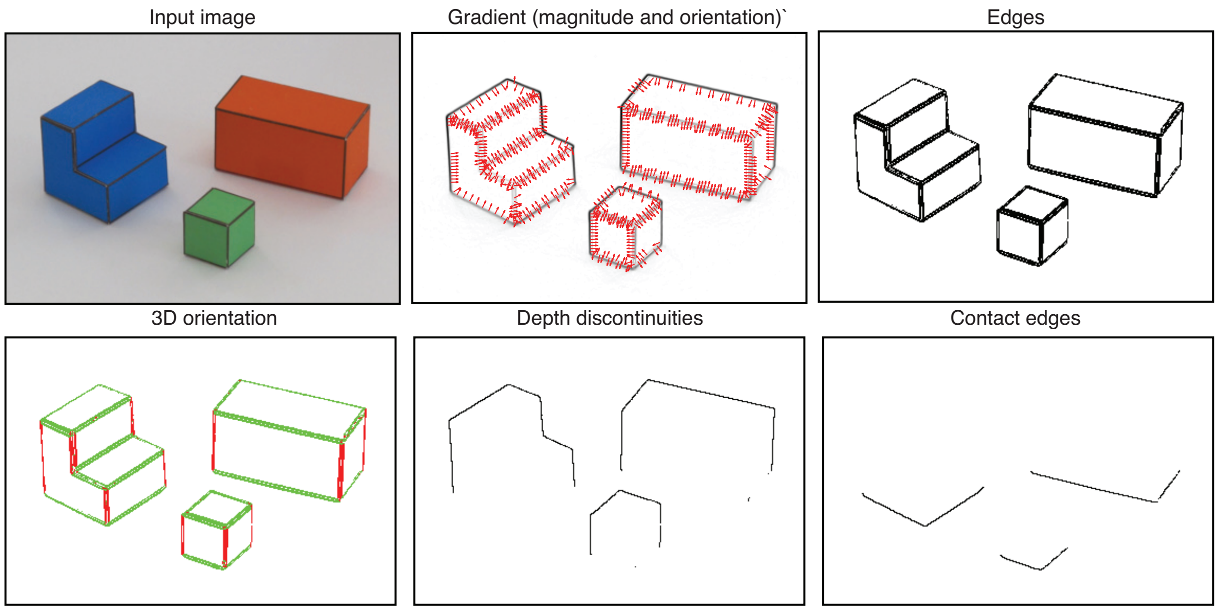

Figure 1.8: Gradient and edge types.

1.5.1 Figure/ground segmentation

In this simple world deciding when 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 challenging task that we will study in lecture 9.

If we assume that the background corresponds to an horizontal ground plane, then, for all pixels that belong to the ground we can set $Y(x, y) = 0$.

1.5.2 Occlusion edges

An occlusion boundary separates two different surfaces at different distances from the observer.

In an occlusion edge, it is also important to know which object is in front as this will be the one owning the boundary. Knowing who owns the boundary is important as an edge provides cues about the 3D geometry, but those cues only apply to the surface that owns the boundary.

In this simple world, we will assume that objects do not occlude each other (this can be relaxed) and that the only occlusion boundaries are the boundaries between the objects and the ground. However, as we next describe, not all boundaries between the objects and the ground correspond to depth gradients.

1.5.3 Contact edges

Contact edges are boundaries between two distinct objects but where there exists no depth discontinuity. Despite that there is not depth discontinuity, there is an occlusion here (as one surface is hidden behind another), and the edge shape is only owned by one of the two surfaces.

In this simple world, if we assume that all the objects rest on the ground plane, then we can set $Y(x, y) = 0$ on the contact edges.

Contact edges can be detected as transitions between object (above) and ground (below). In the simple world only horizontal edges can be contact edges. We will discuss next how to classify edges according to their 3D orientation.

1.5.4 Generic view and non-accidental scene properties

Despite that in the projection of world coordinates to image coordinates we have lost a great deal of information, there are a number of properties that will remain invariant and can help us in interpreting the image. Here is a list of some of those invariant properties (we will discuss some of them in depth in lecture 11 when talking about camera geometry):

- Collinearity: a straight 3D line will project into a straight line in the image.
- Cotermination: if two or more 3D lines terminate at the same point, the corresponding projections will also terminate at a common point.
- Intersection: if two 3D lines intersect at a point, the projection will result in two intersecting lines
- Parallelism: (under weak perspective)
- Symmetry (under weak perspective)
- Smoothness: a smooth 3D curve will project into a smooth 2D curve.

Note that those invariances refer to the process of going from world coordinates to image coordinates. The opposite might not be true. For instance, a straight line in the image could correspond to a curved line in the 3D world but that happens to be precisely aligned with respect to the viewers point of view to appear as a straight line. Also, two lines that intersect in the image plane could be disjoint in the 3D space.

However, some of these properties (not all), while not always true, can nonetheless be used to reliably infer something about the 3D world using a single 2D image as input. For instance, if two lines coterminate in the image, then, one can conclude that it is very likely that they also touch each other in 3D. If the 3D lines do not touch each other, then it will require a very specific alignment between the observer and the lines for them to appear to coterminate in the image. Therefore, one can safely conclude that the lines might also touch in 3D.

These properties are called *non-accidental properties* as they will only be observed in the image if they also exist in the world or by accidental alignments between the observer and scene structures. Under a *generic view*, nonaccidental properties will be shared by the image and the 3D world.

Let's see how this idea applies to our simple world. In this simple world all 3D edges are either vertical or horizontal. Under parallel projection, we will assume that 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, in general, 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 1.9, 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 occur, and a more general algorithm will need to account for that. But for the purposes of this chapter we will consider images with generic views only.

In figure 1.8 we show the edges classified as vertical or horizontal using the edge angle. Anything that deviates from 2D verticality in 15 degrees is labeled as 3D horizontal.

We can now translate the inferred 3D edge orientation into linear constraints on the global 3D structure. We will formulate these constraints in terms of $Y(x, y)$. Once $Y(x, y)$ is recovered we can also recover $Z(x, y)$ from equation 1.3.

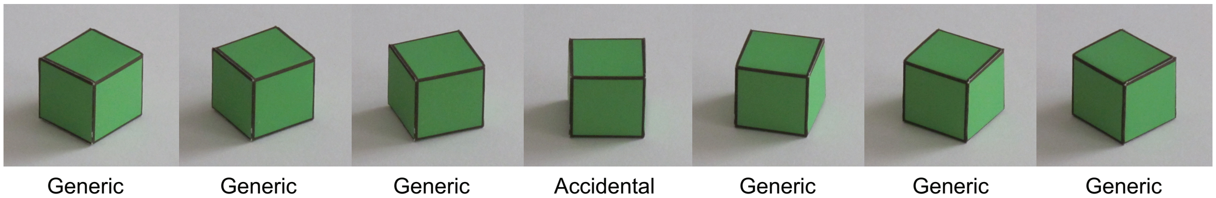

Figure 1.9: Generic views of the cube allow 2d vertical edges to be classified as 3d vertical edges, and colinear 2d edge fragments to be grouped into a common 3d edge. Those assumptions break down only for the center image, where the cube happens to be precisely aligned with the camera axis of projection.

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

$$\partial Y/\partial y = 1/\cos(\theta) \tag{1.11}$$

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

$$\partial Y/\partial \mathbf{t} = 0 \tag{1.12}$$

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:

$$\partial Y/\partial \mathbf{t} = \nabla Y \times \mathbf{t} = -n_y \partial Y/\partial x + n_x \partial Y/\partial y \tag{1.13}$$

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.

1.5.5 Constraint propagation

Most of the image consists of flat regions where we do not have such edge constraints and we thus don't have enough local information to infer the surface orientation. Therefore, we need some criteria in order to propagate information from the boundaries, where we do have information about the 3d structure. This problem is common in many visual domains.

In this case we will assume that the object faces are planar. Thus, flat image regions impose these constraints on the local 3D structure:

$$\partial^2 Y/\partial x^2 = 0 \tag{1.14}$$
$$\partial^2 Y/\partial y^2 = 0 \tag{1.15}$$
$$\partial^2 Y/\partial y \partial x = 0 \tag{1.16}$$

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 $\partial^2 Y/\partial y^2$

1.5.6 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:

$$a_i \mathbf{Y} = \mathbf{b_i} \tag{1.17}$$

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

We can translate all the constraints 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 following cost function:

$$J = \sum_i (a_i \mathbf{Y} - \mathbf{b_i})^{\mathbf{2}} \tag{1.18}$$

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

$$J = \sum_i w_i (a_i \mathbf{Y} - \mathbf{b_i})^{\mathbf{2}} \tag{1.19}$$

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

$$\mathbf{AY} = \mathbf{b} \tag{1.20}$$

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 1.10 shows the resulting 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).

A few things to reflect on:

- It works. At least it seems to work pretty well. Knowing how well it works will require having some way of evaluating performance. This will be important.
- But it can not possibly work all the time. We have made lots of assumptions that will work only in this simple world. The rest of the book will involve upgrading this approach to apply to more general input images.

Despite that this approach will not work on general images, many of the generals ideas will carry over to more sophisticated solutions (e.g., gather and propagate local evidence). One thing to think about is: if information about 3D orientation is given in the edges, how does that information propagate to the flat areas of the image in order to produce the correct solution in those regions?

Evaluation of performance is a very important topic. Here, one simple way of visually verify that the solution is correct is to render the objets under new view points. Figure 1.11 shows the reconstructed 3D scene rendered under different view points.

We can also run the algorithm with shapes that do not satisfy the assumptions that we have made for the simple world. Figure 1.12 shows the *impossible steps* figure from [1]. Figure 1.12 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.

1.6 Recognition

Despite of having a 3D representation of the structure of the scene, the system is still unaware of the fact that the scene is composed by a set of distinct objects. For instance, as the system lacks a representation of which objects are actually present in the scene, we can not visualize the occluded parts. The system can not do simple tasks like counting the number of cubes.

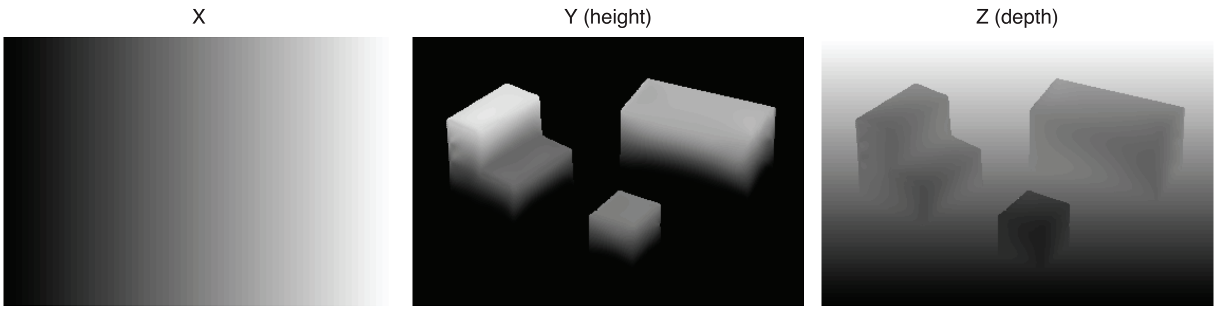

Figure 1.10: The solution to our vision problem: for the input image of fig 1.7, the assumptions and inference scheme described below leads to these estimates of 3d world coordinates at each image location. Note that the world coordinates X, Y, and Z are shown as images.

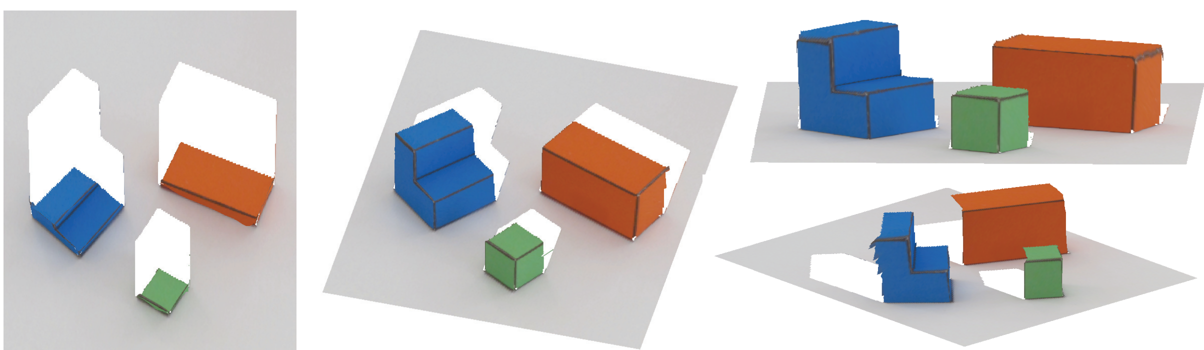

Figure 1.11: To show that the algorithm for 3D interpretation gives reasonable results, we can re-render the inferred 3D structure from different viewpoints, the rendered images show that the 3D structure has been accurately captured.

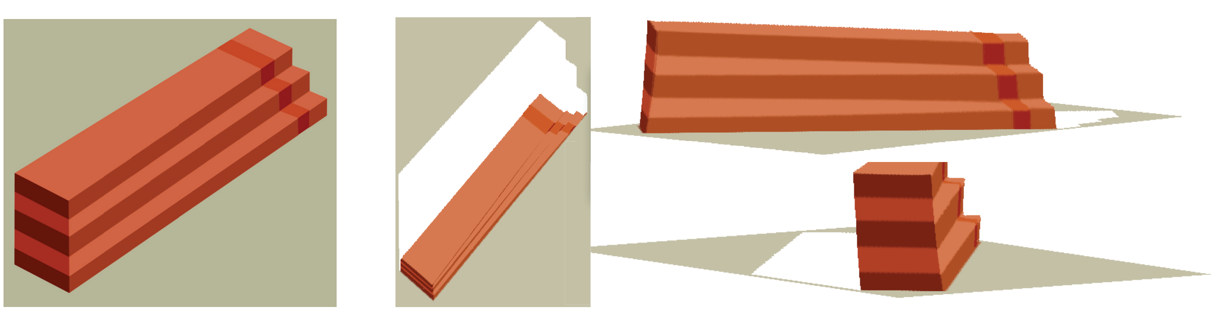

Figure 1.12: Reconstruction of an impossible figure (inspired from [1]): the algorithm does the best it can. Note how the reconstruction seems to agree ask with how we perceive the impossible image.

A different approach to the one discussed here is model based scene interpretation (chapters 20-24) where we could have a set of predefined models of the objects that can be present in the scene and the system should try to decide if they are present or not in the image, and recover their parameters (pose, color, etc.)

Recognition allows indexing properties that are not directly available in the image. For instance, we can infer the shape of the invisible surfaces. Recognizing each geometric figure also implies extracting the parameters of each figure (pose, size, ...).

Bibliography

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

[2] Edward H. Adelson and James R. Bergen. The plenoptic function and the elements of early vision. In Michael S. Landy and Anthony J. Movshon, editors, *Computational Models of Visual Processing*, pages 3–20. MIT Press, Cambridge, MA, 1991.

[3] P. Cavanagh. Vision is getting easier every day. *Perception*, 24:1227–1232, 1996.

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

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