Journal of Mathematical Psychology 56 (2012) 217–231

Contents lists available at SciVerse ScienceDirect

Journal of Mathematical Psychology

journal homepage: www.elsevier.com/locate/jmp

Tutorial

A tutorial explaining a machine vision model that emulates human performance when it recovers natural 3D scenes from 2D images

Yunfeng Li[a,*], Tadamasa Sawada[a], Longin Jan Latecki[b], Robert M. Steinman[a], Zygmunt Pizlo[a]

[a] *Department of Psychological Sciences, Purdue University, United States*
[b] *Department of Computer and Information Sciences, Temple University, United States*

ARTICLE INFO

Article history:
Received 10 October 2011
Received in revised form
8 June 2012
Available online 4 August 2012

Keywords:
Natural scene recovery
Simplicity constraints

ABSTRACT

This tutorial has two purposes: (i) it provides an example of a computational model that emulates what human beings see in natural environments, and (ii) it explains why, as well as how, each computation can be done.

Contents

0. Introduction

This paper presents a step-by-step description of our computational model of 3D shape and scene recovery. This kind of presentation was used to make it capable of serving as a tutorial for interested readers. It is addressed primarily to psychologists, not to machine vision researchers despite the fact that it resembles a machine vision application. There are two obvious reasons why it will be of interest to psychologists but not to machine vision researchers. First, the machine vision, as well as the robotics, community is developing computational tools for object and scene analysis that lie outside of the context of biological vision. Most of the tools they use cannot be used to model human vision. They use non-biological sensors, such as lasers or structured light, or they try to solve 2D vision problems, not the 3D problems facing human beings and any machine that tries to emulate them. We model *vision* initiated by two cameras that are sensitive to the *visible* portion of the electromagnetic radiant energy spectrum with the goal of achieving a veridical recovery of 3D shapes and 3D scenes. The machine vision and robotics communities have different goals and use different tools to achieve them. Second, the performance of all existing machine vision systems in object reconstruction and recognition tasks does not even come close to human performance. Human performance is so much better than the performance of any machine vision system available today that these systems are of little, if any, interest to psychologists. For example, state-of-the-art machine vision systems can detect a person in an image 51.6% of the time and a chair 23.2% of the time, according to the most recent results of the PASCAL Visual Object Challenge, http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2011/results/index.html. We humans detect people and chairs 100% of the time. This number,

[*] Corresponding author.
E-mail address: li135@purdue.edu (Y. Li).

100%, is not even in the current vocabulary of the machine vision community much less descriptive of the performance of any vision machine. Those of us who talk to chairs and sit on people fair poorly in everyday life. If a computational model cannot, at least in principle, guarantee nearly perfect performance, it can be rejected as a model of human vision. The computational model described in this tutorial is the first successful attempt to explain the human beings' perception of real 3D shapes and 3D scenes based only on information provided by one or two real 2D images. Our model builds on several results and tools developed in the machine vision community, but the particular choices about which tools should be used and how they should be used were, in all cases, dictated by the psychological plausibility of the computations. Some of the ideas and methods presented here were developed by us. We indicate the relevant sources whenever we used ideas or tools developed by others.

The basic idea underlying our new approach for recovering natural 3D scenes makes use of mechanisms employed by human beings to recover 3D shapes from 2D images (Li, 2009; Li, Pizlo, & Steinman, 2009; Pizlo, 2008; Pizlo, Sawada, Li, Kropatsch, & Steinman, 2010; Sawada, 2010; Sawada & Pizlo, 2008). The novel, critical aspect of our approach is that *a priori* constraints are at least as important in 3D vision as visual data. This approach is different from all other, more conventional approaches in which the reconstruction of 3D shapes and 3D scenes is a hierarchical process based on a number of independent visual modules responsible for acquiring and combining pieces of visual information, called "depth cues", e.g., texture, shading, motion, disparity, and vergence. Our novel approach, described with David Marr's (1982) widely-known terminology, bypasses a viewer-centered representation by recovering geometrical properties of objects and their environment in an object-centered representation. Our approach is preferable to the conventional approach for two different, but related reasons. First, it is known that despite the fact that the human beings' perception of 3D distances and 3D sizes is usually *not* veridical, the human being's perception of 3D shapes *is always* veridical (Li, Sawada, Shi, Kwon, & Pizlo, 2011).[1] Second, several very effective *a priori* constraints for the perception of 3D shape are known, but no effective constraints are known for the perception of the 3D distances between pairs of 3D points. An additional advantage inheres in starting to recover 3D scenes with 3D shapes. Starting with 3D shapes allows the observer to "see" the "invisible" back parts of opaque objects. Paraphrasing Bartlett (1932) to put our novel approach into a broad historical perspective, we can say that using 3D shapes to construct 3D scenes allows an observer to actually "go beyond the information given". The special role and significance of shape in visual perception was appreciated and highlighted by the Gestalt Psychologists almost 100 years ago, but the mathematical and computational tools necessary to formulate their ideas and to make use of them in computational models did not become available until recently. This tutorial reviews several of these tools and illustrates how they can be applied to the recovery of a natural 3D scene, like the scene shown in Fig. 1.

We prepared this tutorial because we believe that, at present, the best way to test any theory in vision is to implement a computational model of the underlying perceptual mechanisms and to show that it can be used effectively by an autonomous robotic system. This kind of robotic system acquires visual information, and then plans and executes actions without any intervention by the designer of the system. We have succeeded in developing a computational model that does this. It recovers a 3D object from one of its 2D images and then uses these mechanisms to recover naturalistic 3D scenes. These scenes permit the robot to perform complex navigations without any aid from a human being, the kind of activity previously only performed by an alert, human being.

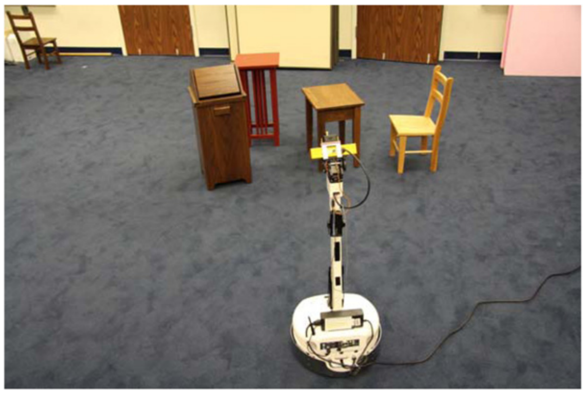
Fig. 1. Five man-made 3D objects within a natural scene and our robot (Čapek) looking at them (the room used for robot navigation). A camera mounted on the top of this robot provides the visual information used to guide its navigations. An inclinometer, mounted on top of the camera, provides the robot with the information it uses about the direction of gravity.

Note that ours is not the approach most often used by others working in vision today. Vision researchers, more often than not, simply state what they call a "theory" in plain English and go on to describe some qualitative aspects of what they believe to be a potential perceptual mechanism. Even when a contemporary vision researcher has actually used a mathematical or computational model, more often than not, it was tested with synthetic images or only with a few hand-picked real images. Such tentative and partial approaches to providing an explanation of a limited visual process was justifiable some years ago but we believe that we have reached the point at which vision researchers should be much more ambitious. A nearly complete, working theory of at least some particularly significant aspect of visual processing should be provided now that this is no longer beyond reach. Meeting such expectations can be best served by verifying that any current theory meets at least two criteria, namely, that: (i) it is relatively complete, and (ii) it has no implicit or unjustified explicit assumptions. The best way to do this is to build a machine that can actually see as we do, a more complex act than is commonly assumed.

Traditionally, the visual and the motor systems have been treated as separate, independent modules that could be and were studied separately. But the fact that they are not nearly independent and that they can, and should be, studied as they work together, rather than separately, was emphasized by Dewey (1896) more than 100 years ago. He emphasized that perception is not, and should not be treated as, a passive process. Perception is only one part of a closed, continuously active "reflex arc", actually more like a circle or a loop, called the "perception–action cycle", in which perception and motor action follow one another and interact continuously in everyday life. This observation is not only very old, it was picked up, promulgated, mulled over and elaborated by many others since it was proposed, most notably by Hebb (1949) and by Gibson (1966), who attempted to include this interaction in their "verbal models" of what is now often referred to as "ecologically-valid vision". Note that the integration of vision and action, originally proposed by Dewey, who credits James (1890) for pointing him in this direction, makes a lot of sense, despite its neglect in most contemporary work, for at least two

[1] By "veridical" we mean that the percept of the shape of an object agrees with its shape in the real world. Note that we are ignoring laboratory experiments that were designed specifically to demonstrate the failure of shape veridicality by using degenerate shapes and/or degenerate viewing directions.

reasons. First, an observer must actively seek information about the environment by using more than one viewing direction. The observer must do this to determine whether his initial viewing direction provided all of the information required to perceive the scene veridically. Second, the observer needs veridical information about the environment because it is essential for the efficient planning and successful execution of the specific behavioral acts that will achieve the desired goals. We recently showed that human observers can reconstruct the 3D shape of an indoor scene, like the scene shown in Fig. 1, from a single viewing position very accurately and precisely (Kwon, 2012). Binocular performance was slightly better than monocular performance, and there was no evidence of systematic distortions of visual space. The reader will surely agree that the best, perhaps even the only, way to study such complex, but very natural, interactive processing is to develop a model that sees and acts, as the human being does, and furthermore, that this model should be realized in the form of a mobile robot now that they are available "off-the-shelf". Note that this proposal is not novel. It was first proposed long before its implementation was viewed as either possible or imminent. It goes back at least to Richard Feynman, who is reported as having said "what I cannot create I don't understand" and/or was proposed at the early stages of cognitive science by Miller, Galanter, and Pribram (1960) who said that "The creation of a model is proof of the clarity of the vision. If you understand how a thing works well enough to build your own, then your understanding must be nearly perfect (p. 46)". So, all we are really saying here, is that it is time to put these words into practice. We prepared this tutorial to make it easier for individuals, who share this goal, to implement vision theories in a machine.

The tutorial begins with a description of why, as well as how, a camera should be calibrated when it is used for "machine vision". This is followed by a description of the most important features of our 3D shape recovery model. These features will be presented with special emphasis on the nature and role of our model's *a priori* constraints. Next, problems inherent in the recovery of the shape of a 3D scene are stated and possible methods of solving these problems are described. The solutions call attention to the importance of non-visual, as well as to visual, *a priori* constraints. Note that by recovering the shape of a 3D scene, we mean recovering the geometry of the 3D scene up to one unknown parameter, namely, the overall scale of the scene and the objects within it. The scale can be based on the estimation of a *single* distance in a 3D scene. The height of the observer, a human or a robot, can serve as a particularly useful single distance for a potential navigator. Finally, having explained what it takes to recover 3D shapes and then to use this information to recover a 3D scene, we will describe the first step in visual processing, which is usually referred to as Figure-Ground Organization (FGO), the terminology favored by the Gestalt Psychologists a century ago. Discussing FGO, the very first stage of visual processing, last will allow us to actually specify what the output of the first stage of visual processing should be like. Prior to our new approach to the recovery of 3D shapes and 3D scenes, there was not even a good definition of what Figure-Ground Organization was, much less of what Figure-Ground Organization should accomplish or how this could be done.

In our approach, FGO refers to (i) finding the 2D region and its occluding contour for each object in the 2D image, as well as to (ii) finding the 3D region and its bounding box in the 3D scene "out there", where the object resides. Once this is done, the internal contours required for 3D shape recovery are extracted and the 2D information about the 3D symmetry of the shape and the planarity of its contours are determined and included in the description of the "figure" by labeling its contours. It is clear that in our approach, FGO refers both to properties of the 2D image and to properties of the 3D shape. This differs from the conventional view of FGO that limits FGO to 2D features. Note that our claim that FGO includes 3D, as well as 2D features, was suggested by the Gestalt Psychologists when they emphasized that the organized figure is always perceived in front of its background, which is perceived as lying behind as well as around the figure.

1. Calibration of the camera

The order of the next sections does not reflect the order of the computations by the visual system or the order of importance of our computations. The order simply reflects what we believe to be the best and simplest way of explaining what we did and why we did it. Why it is important to calibrate your camera if you want to use it in research on visual perception will be explained below. We start with the definition of a "camera matrix". Most of the equations presented in this paper use this matrix. The process of estimating a camera's matrix is called "camera calibration". A camera matrix is a 3 by 4 matrix that defines the geometric properties of a camera, like its focal length, principal point, etc. These properties characterize the perspective projection from a 3D scene to a 2D image. Why is the camera matrix important? Consider a psychophysical experiment on the perception of objects from perspective images, such as photographs. The geometrical properties of these photographs must be known if the experimenter wants to show photographs of the objects to his subject. Specifically, the subject's eye (more precisely, the center of the perspective projection of the eye, called its "nodal point") must be placed at the center of the perspective projection for any given picture. This must be done if you want the retinal image in the subject's eye produced by the *perspective* photograph of the 3D object to be a *perspective* image of the 3D object. This is how the first demonstration of the rules of perspective projection was done almost 600 years ago by Brunelleschi (see Kubovy, 1986). This method is still used by modern students of vision to set up their experiments, e.g., Attneave and Frost (1969). If the eye is placed at any other point than at the center of the perspective projection, the retinal image produced by a perspective photograph of a 3D object will not be a valid perspective image of this object. Instead, it will be an image of a 3D *projective* transformation of the object because a perspective picture of a perspective picture is not, itself, a perspective picture (Pizlo, 2008). Failing to use a *perspective* projection will almost always lead to a non-veridical percept of the 3D object (see Kubovy, 1986; Pirenne, 1970, for examples of such distortions). The bottom line is that *when 2D perspective images are used for making 3D inferences, the parameters of the camera that took the images must be known.*

Now that you appreciate why a camera matrix is important, we will describe it in detail. Consider the relation between a 3D point V^* in front of a camera and expressed by homogeneous coordinates $(V_X^*, V_Y^*, V_Z^*, V_W^*)^T$ and its 2D camera image v^* also expressed by homogenous coordinates $(v_x^*, v_y^*, v_w^*)^T$. Eq. (1) represents the perspective transformation from V^* to v^*:

$$v^* = KQV^* \tag{1}$$

where

$$K = \begin{pmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad Q = \begin{pmatrix} R_{3x3} & -R_{3x3}C_{3x1} \end{pmatrix}. \tag{2}$$

Matrix K is called the "intrinsic matrix". It defines the camera's intrinsic properties, such as its focal length. Matrix Q is called the "extrinsic matrix". It represents the transformation from the world coordinate system to the camera coordinate system. This transformation consists of a 3D translation $(-C_{3x1})$ followed by a

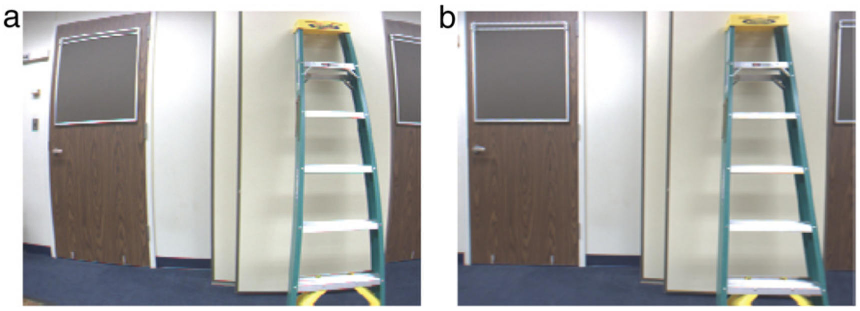

Fig. 2. (a) Radial distortion of a wide angle camera. (b) The image from (a) after calibration.

3D rotation (R_{3x3}) (refer to Faugeras' book (2001) for the details of the camera matrix). The geometric details of R and C are described in the following paragraphs. The product of the intrinsic matrix (K) and the extrinsic matrix (Q) is called the "camera matrix" (P), which is a 3 by 4 matrix

$$P = KQ. \tag{3}$$

Combining Eqs. (3) and (2), we obtain

$$P = \begin{pmatrix} KR & -KRC \end{pmatrix}. \tag{4}$$

In Eq. (1), the 3D point and its image are expressed by homogeneous coordinates because they allow expressing a non-linear perspective projection by using matrix notation. The transformation between the Euclidean coordinates and the homogenous coordinates for a 3D point $(V_X, V_Y, V_Z)^T$ and a 2D point $(v_x, v_y)^T$ is expressed as follows:

$$\begin{pmatrix} V_X & V_Y & V_Z \end{pmatrix}^T = \begin{pmatrix} V_X^* & V_Y^* & V_Z^* \end{pmatrix}^T / V_W^* \tag{5}$$

$$\begin{pmatrix} v_x & v_y \end{pmatrix}^T = \begin{pmatrix} v_x^* & v_y^* \end{pmatrix}^T / v_w^* \tag{6}$$

assuming that V_W^* and v_w^* are not equal to zero.[2] Eqs. (5) and (6) imply that the homogenous coordinates of a point are not unique. For example, $(1, 2, 3, 1)^T$ and $(2, 4, 6, 2)^T$ represent the same 3D point. In practical applications, V_W^* can usually be set to 1. This way, 3D homogenous coordinates are trivially obtained from 3D Euclidean coordinates (and vice versa). The representation of a 3D plane in homogenous coordinates is the same as that of a 3D point: both are four-element vectors. For example, if π^* is a 3D plane, then all points V^* on π^* satisfy $\pi^{*T} V^* = 0$. Similarly, in a 2D image, the representation of points and lines in homogenous coordinates are the same, and they are three-element vectors.

In this paper, some equations use both homogenous and Euclidean coordinates. To avoid confusion, symbols *with asterisks* represent the homogenous coordinates of geometric primitives, like points, lines, or planes. The symbols *without asterisks* represent the Euclidean coordinates. The individual parameters in the camera matrix are described next.

Consider the parameters in the intrinsic camera matrix (K). $(\mu_0, \nu_0)^T$ is the principal point of a camera, the point of intersection of the camera image plane with a line emanating from the center of a perspective projection that is orthogonal to this plane (see Fig. 3). The principal point is close to the center of the camera image, but it is never exactly at the center, due to technical limitations inherent in the design of the camera. Why are there such limitations? Consider the fact that when the physical size of a camera's image, containing 2000 by 3000 pixels, is less than 1×1 cm^2, the displacement of the center of the camera lens by as little as one millimeter translates into a displacement of the principal point by more than 100 pixels. This makes it almost impossible to mount the lens exactly in front of the center of the camera's image. The green dot in Fig. 5 represents the center of the picture and the white dot represents the principal point of the camera that was used to make most of the examples included in this paper. In our setup, this point is displaced from the center of the image by about 0.6°. In the human eye, the principal point corresponds to a region of best vision near the center of the retina, called the "fovea". The fovea is well-defined both anatomically and perceptually. Anatomically, the flat floor of the fovea is a disc with a diameter of about 1.5°. It contains only receptors, called "cones". Perceptually, the fovea serves as the center of the visual field. It is the region in which detail vision is most acute. When an observer orients his eye to look directly at a feature in order to examine its details, the eye's orientation will cause the feature's retinal image to fall near the center of the fovea, where it is said to be "fixated". An observer can maintain fixation of an attended object with high precision: the standard deviation of eye position during maintained fixation is only 3 or 4 min of arc (Steinman, 1965). This is equivalent to only two pixels in a camera whose field of view is 60° and whose image is an array of 2000 by 3000 pixels.

The next intrinsic camera parameter considered is called its "focal distance". The focal distance is the distance between the center of a perspective projection and the camera image (Fig. 3(b)). In the intrinsic camera matrix, the focal distance is defined in terms of the number of pixels along the X axis (α_x) and Y axis (α_y) in the camera coordinate system. In other words, both the interval of length α_x pixels along the X axis and the interval of length α_y pixels along the Y axis are equal to the focal length. In modern cameras, the difference between α_x and α_y is very small, so we can assume that they are equal. In the human eye, the focal distance is about 2 cm, which is the approximate diameter of the human eyeball. The third intrinsic parameter of a camera is called the "skew" (s) which specifies how much a pixel is biased from a perfect rectangle. For most modern cameras, the skew parameter is zero.

There is one more intrinsic camera parameter. It measures what is called its "radial distortion". This parameter is not expressed in Eq. (1) because the radial distortion cannot be represented as a linear transformation. In an image with radial distortion, the straight lines at the periphery tend to be curved (see Fig. 2). Radial distortion is obvious with cameras whose lenses have a large field of view (e.g., 60°) or a small focal length.

Now consider the extrinsic matrix Q. It defines the transformation between the world coordinate system and the camera coordinate system. The specification of the world coordinate system depends on the application. For example, if the task is to recover a 3D scene in a room, it is natural to use one of the corners of the room as the origin and the three edges of the room emanating from this corner as the X, Y and Z axes. The camera coordinate system does not actually depend on the application. It is a fixed characteristic of the camera. This coordinate system is defined as follows:

[2] In homogenous coordinates, if the last value of a vector is equal to zero, the vector represents a point at infinity.

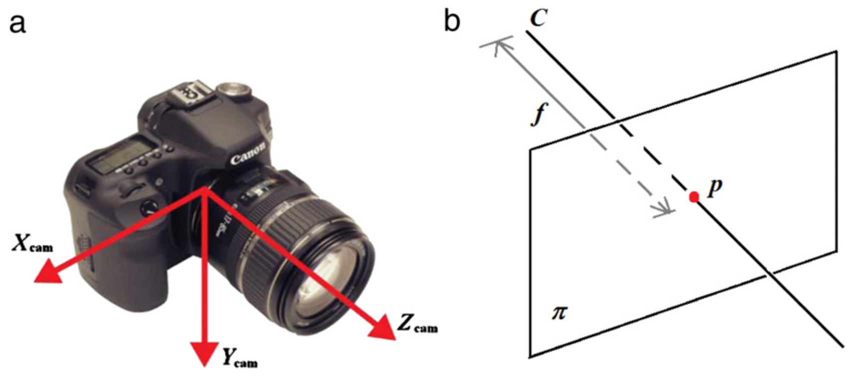

Fig. 3. (a) The camera's coordinate system. (b) Schematic illustration of a camera. C represents the projection center. p is the principal point. The line pC is orthogonal to the image plane π. f represents the camera's focal length.

the origin is the center of perspective projection of the camera. The XY plane is parallel to the camera image plane. The X axis coincides with the X axis of the camera image. The Z axis represents the depth direction (see Fig. 3(a)). The vector C in Q is the projection center expressed in the world coordinates. R_{3X3}, a rotation matrix, represents the orientation of the camera coordinate system. Specifically, the three row vectors in R correspond to the directions of the X, Y and Z axes of the camera coordinate system expressed in the world coordinate system.

It follows that the camera matrix not only includes a camera's properties, it also includes its orientation and position in the environment (see Faugeras, 2001, and Hartley & Zisserman, 2003, for additional details about the camera matrix). The process of estimating camera parameters is called "camera calibration". A camera is calibrated by acquiring multiple images of a reference scene, whose geometry is known. Once the 3D coordinates of the scene and the 2D coordinates in its image are known, one can solve for the camera's unknown intrinsic and extrinsic parameters. Open access software for calibrating a camera, can be found at (e.g., OpenCV: http://opencv.willowgarage.com/wiki/FullOpenCVWiki).

2. Recovery of a 3D shape and a 3D scene

The camera matrix defines a camera's geometry—it specifies how to project a 3D point onto a 2D image plane. Given a 3D point, its image is uniquely determined. Therefore, generating a 2D image from a 3D scene is an easy "forward problem" (Pizlo, 2001; Poggio, Torre, & Koch, 1985). However, the "inverse problem", recovering a 3D scene from its 2D image, is difficult because the solution is not unique, i.e., for any given 2D image point, there are infinitely many 3D points that can produce the same 2D image point. Inverse problems are almost always difficult because they are "ill-posed" and "ill-conditioned". In plain English, inverse problems are "insoluble". The only way to solve an inverse problem is to impose *a priori* constraints on the family of possible interpretations, and then combine these constraints with the available data to find the most reasonable solution. Ideally, it will be the correct, veridical, interpretation of the conditions in the physical world. Fig. 4 shows a flow chart illustrating how to recover a 3D scene from a single 2D image based on the *a priori* constraints, namely, from the direction of gravity, symmetry and planarity (the pseudo code is provided in Appendix C). The model is described in detail just below.

The *symmetry* of the 3D shape is a strong *a priori* constraint. Given a 2D perspective image of a symmetrical 3D shape, its symmetrical 3D interpretation is unique except for the size and position. The following equations show how to use the camera matrix (P) to recover a pair of 3D symmetric points (X_1 and X_2) from their 2D image (x_1 and x_2) if the vanishing point v for the line connecting X_1 and X_2 is given.[3]

From Eq. (4), the camera matrix P can be expressed as follows

$$P = \begin{pmatrix} M_{3X3} & p_4 \end{pmatrix} \tag{7}$$

where $M_{3X3} = KR$ and $p_4 = -KRC$. Let x_1 and x_2 be expressed by the Euclidean coordinates and x_1^* and x_2^* be their homogenous coordinates with the third element equal to 1. Then the set of all 3D points whose image is x_1 (or x_2) can be expressed as follows (Eq. (6.14) in Hartley & Zisserman, 2003):

$$X_i = M^{-1}(k_i x_i^* - p_4) \quad i = 1, 2 \tag{8}$$

k_i are free parameters. For the solutions X_1 and X_2 to be symmetrical, k_1 and k_2 must satisfy the following equation (refer to Appendix A for the derivation):

$$\begin{pmatrix} v^{*T}M^{-T}M^{-1}x_1^* & v^{*T}M^{-T}M^{-1}x_2^* \\ |x_1 - v| & -|x_2 - v| \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 2v^{*T}M^{-T}M^{-1}p_4 - 2d \\ 0 \end{pmatrix}. \tag{9}$$

For the recovered pairs of symmetrical 3D points, the normal of the symmetry plane is determined by the vanishing point v, and is equal to $M^{-1}v^*$. The position of the symmetry plane is determined by the parameter d. d is a free parameter and it can be any real number, which determines the size (or position) of a recovered 3D object. Fig. 7(a) shows five objects recovered from the same 2D camera image. Their symmetry planes have the same orientation, but different positions. The recovered object is small when it is close to the camera (the cyan box in Fig. 7(a)). The recovered object is large when it is far from the camera.

Although Eq. (9) looks complex, it can be simplified in applications after making some assumptions about the camera's parameters. For example, if the skew s is equal to 0, α_x and α_y are identical, and the origin of an image coincides with the principal point, then K is a diagonal matrix ($K = \text{diag}(\alpha_x, \alpha_x, 1)$). Furthermore, if the world coordinate system coincides with the camera coordinate system, then R is an identity matrix and p_4 is a zero vector. It follows that $M = K$.

Eq. (9) suggests that in order to recover a symmetrical 3D shape, the visual system needs to: (1) know where the vanishing point is and (2) establish which points in a 2D image, are the images of pairs of 3D symmetrical points. In order to accomplish these two

[3] The vanishing point in the image is the intersection of the lines connecting the images of pairs of 3D symmetrical points. In 3D space, all of these lines are parallel to one another. In a perspective image, they all intersect at the vanishing point.

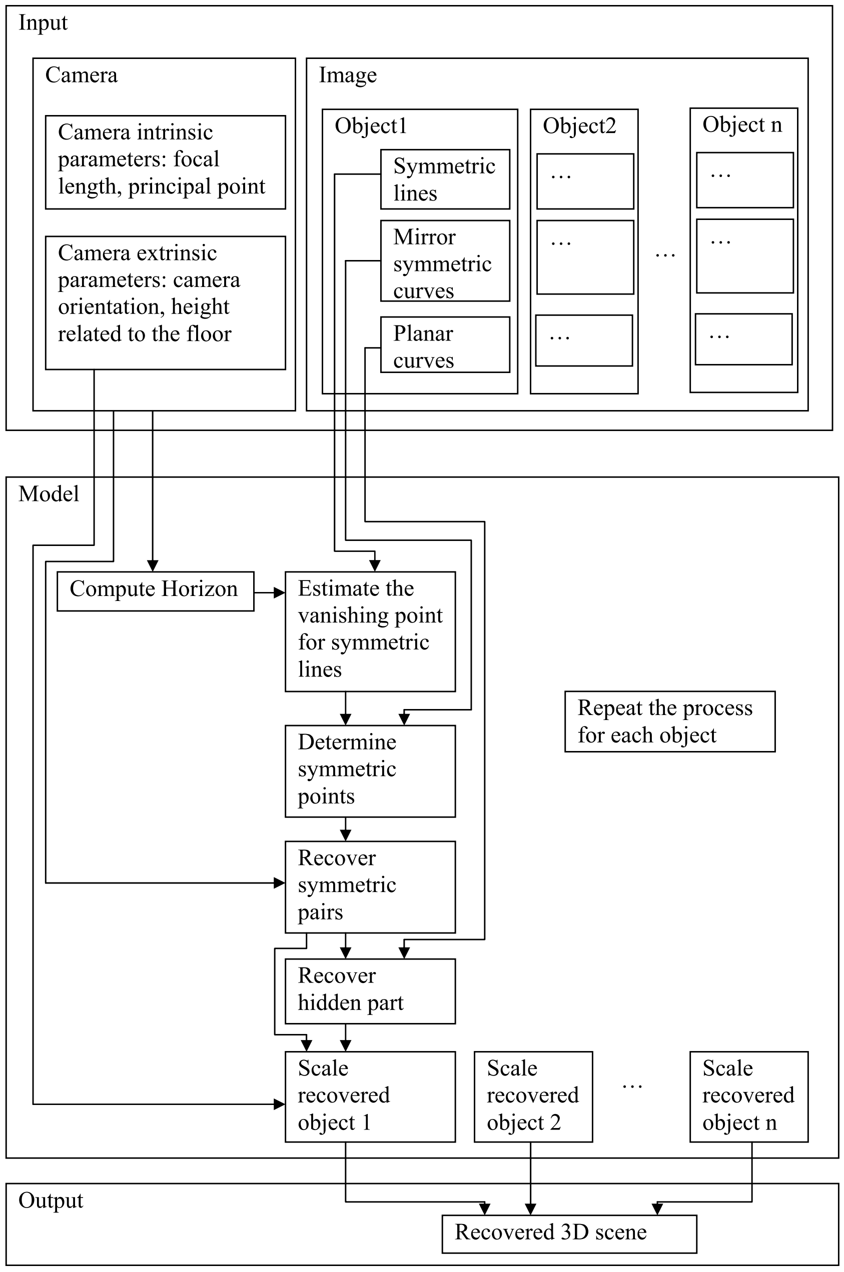

Fig. 4. Flow chart of the model, recovering a 3D scene from a single 2D image.

things, two additional *a priori* constraints will be required, namely, the direction of gravity and a line representing the horizon.[4]

Computation of the vanishing point and identification of symmetric pairs

Gravity is one of the most critical constraints operating in our environment. Gravity is not only responsible for stability in our environment; it is also most likely to be responsible for the symmetry of almost all animals' bodies and the design of symmetrical objects. If the ground plane is horizontal, it is orthogonal to the direction of gravity. An object will be stable if its body is symmetrical with respect to the plane parallel to the direction of gravity. For example, a symmetrical animal will not fall on its side when it stands. It follows that given a symmetrical 3D shape standing on a horizontal ground, the line segments connecting the symmetrical points are parallel and orthogonal to gravity. For all parallel lines that are orthogonal to gravity, their vanishing points fall on a horizon. If the ground plane is not horizontal, the symmetry line segments of symmetrical objects standing on the ground are parallel to the ground but not orthogonal to the direction of gravity. The corresponding vanishing line is then determined by the actual ground plane, not by the plane orthogonal to the direction of gravity.

Assume that the normal of the ground floor is N_h in the world coordinate system, then the horizon (i.e., the vanishing line corresponding to the horizontal ground plane) is expressed as follows (Result 8.16 in Hartley & Zisserman, 2003):

$$l_h^* = M^{-T} N_h. \tag{10}$$

For a calibrated camera, the horizon is known *before* the image is taken, which means that this information is truly *a priori*. Existing evidence suggests that the same is true in human vision. The

[4] The horizon is a vanishing line on the image plane, which is a perspective projection of the line at infinity on any plane parallel to the horizontal ground plane.

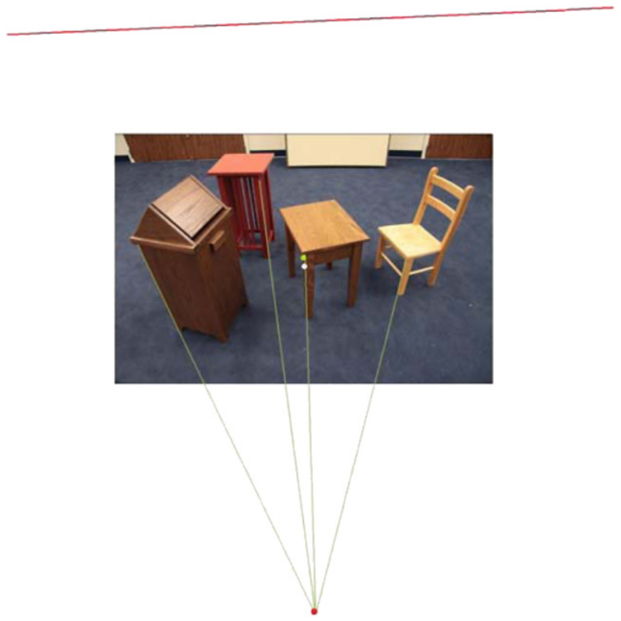

Fig. 5. The red line above the image of a 3D scene shows the horizon in this scene. The red dot below the image shows the vanishing point corresponding to the 3D vertical lines. The principal point is marked by a white dot near the center of the image. The geometrical center of the image is marked by the green dot. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

perception of the horizontal direction is affected by both visual and vestibular cues (Matin & Fox, 1989). Once the horizon is known, we can search for the vanishing point, which is the intersection of the 2D symmetry line segments of a given object. Since the vanishing point must be on the horizon, the search is determined by only one free parameter. Without the horizon, there are two unknown parameters specifying the position of the vanishing point, and the point cannot be estimated reliably (specifically, its distance from the object's image in the 2D camera image). The horizon provides a very strong constraint for this less reliable parameter. Eq. (10) shows that the horizon is equal to the product of the direction of gravity and the inverse of transposed M. Because M (the product of the intrinsic matrix K and the rotation matrix R) is unrelated to the position of the camera (C), the translation of the camera in 3D space leaves the horizon and all vanishing points in the image invariant. These invariant features are likely to be useful in robot navigation.

The vanishing point on the horizon is obtained by computing the intersection between the symmetry lines (the green lines in Fig. 6(b)) and the horizon (the blue line). The green lines (contours in the image) are not always perfectly straight. Therefore, the first step in computing the vanishing point is to approximate (by using least squares) the symmetry lines with straight lines.

Let x_i^* represent the 2D points on a symmetry line. Let $A_{kx3} = \begin{pmatrix} x_1^* & x_2^* & \cdots & x_k^* \end{pmatrix}^T$. Then the approximating line l^* is parallel to the eigenvector of $(A^TA)_{3x3}$ whose corresponding eigenvalue is the smallest.[5] Once each symmetry line is approximated by a straight line, we can estimate the vanishing point for these symmetry lines. Suppose for one object, $l_1^*, l_2^*, \ldots, l_n^*$ are the n symmetry lines. Because of noise, the intersections of symmetry lines with the horizon are not identical. Therefore, we estimate the vanishing point as the point that has the least square distance to all symmetry lines.

Let the horizon $l_h^* = \begin{pmatrix} (l_h^*)_x & (l_h^*)_y & (l_h^*)_w \end{pmatrix}$, then $m_h^* = \begin{pmatrix} -(l_h^*)_y & (l_h^*)_x & 0 \end{pmatrix}$ represents the direction of the l_h^*. Suppose v_0^* is one point on the horizon such that $(v_0^*)^T l_h^* = 0$, then the vanishing point is estimated as:

$$v^* = v_0^* + um_h^* \tag{11}$$

where $u = -\frac{(m_h^*)^T B^T B v_0^*}{(m_h^*)^T B^T B m_h^*}$ and $B = \begin{pmatrix} l_1^* & l_2^* & \cdots & l_k^* \end{pmatrix}^T$. The derivation is given in Appendix B.

Once the vanishing point is estimated, the pairs of points in the 2D image, which are images of symmetrical points of an object, can be established as intersections of pairs of corresponding contours and the lines emanating from the vanishing point. Fig. 6(c) illustrates how to identify the symmetrical pair of curves. These curves are drawn in blue. The symmetrical pairs of points on the corresponding curves are collinear with the vanishing point. The 3D symmetrical shape can then be recovered by recovering all pairs of symmetrical points according to Eq. (8).

Recovery of the hidden part

We just showed how to recover pairs of symmetrical 3D points by using Eq. (8). Note, however, that in order to recover a 3D point, both the image of this point and of its symmetrical counterpart had to be known. In other words, the symmetrical pairs in a 2D image must be visible. For example, the back, the seat and the front legs of the chair in Fig. 6(b) can be recovered on the basis of the symmetry constraint because their corresponding symmetrical contours are visible. However, the two rear legs cannot be recovered by using symmetry alone because one of the legs is hidden. In this case, we begin by using the planarity constraint to recover the point of the chair that is visible. The contours representing the right side of the chair shown in Fig. 6(b) are coplanar (approximately) and we can estimate the plane containing these contours from the points and contours that were recovered by using Eq. (8) (they could be recovered because both symmetrical pairs were visible). Once this is done, the intersection of this plane and the plane defined by the image of the visible right rear leg and the projection center of the camera, is a 3D line containing the recovered right rear leg. Its invisible, symmetrical counterpart is obtained by reflecting the recovered right rear leg with respect to the symmetry plane (see Li et al., 2009, for details).

Recovering the shape and scale of a natural 3D scene

We pointed out (above) that for the recovered 3D shapes, their sizes and their positions are undetermined, but "placing" them on the ground will make it possible to uniquely determine the relative positions, sizes and pair-wise distances among all of the 3D objects. When the shape of a 3D object is recovered, the object can either be small and close to the camera, or large and far from the camera (see Fig. 7(a)). Once the height of a camera above the floor is known, there is *only one* size and *only one* corresponding distance at which a given object will be resting on the floor. For smaller distances, a recovered object would be floating in the air, and for larger distances, the object will be below the floor. Thus, regardless of the number of objects in the scene, their sizes, positions and distances are determined by only one parameter, namely the height of the camera. Ambiguity only remains for objects whose relative position with respect to the floor is unknown: this occurs whenever the bottom part of an object is occluded. Whenever this happens, the size and distance of the object will be uncertain. But, because real objects cannot occupy the same physical space, this uncertainty can be reduced by using information obtained from nearby objects. It is very likely that the *a priori* constraints used by our model are also used by the human visual system. The subject's

[5] The singular value decomposition (SVD) method can be used to find eigenvectors. The matrix A can be decomposed and expressed as $A = USV^T$ where U and V are orthonormal matrices. S is a diagonal matrix and its values are sorted in a descending order. The direction of the approximating line is represented by the last column vector of V.

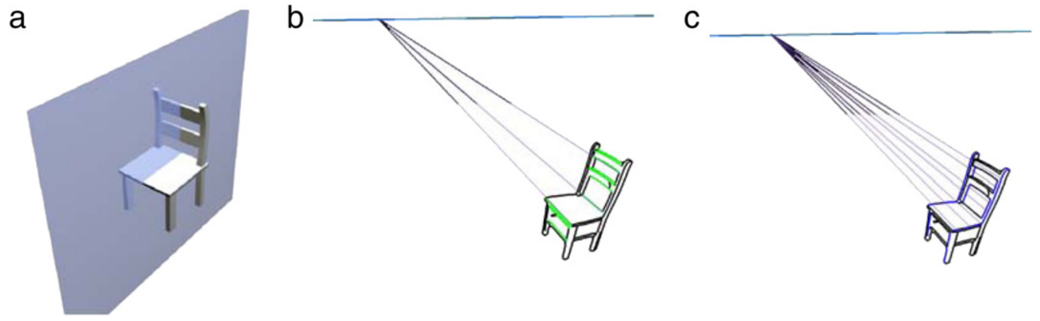

Fig. 6. (a) The symmetry plane of a 3D object resting on the ground is vertical. (b) Symmetry line segments are indicated by green. (c) Pairs of symmetrical contours are marked by blue. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

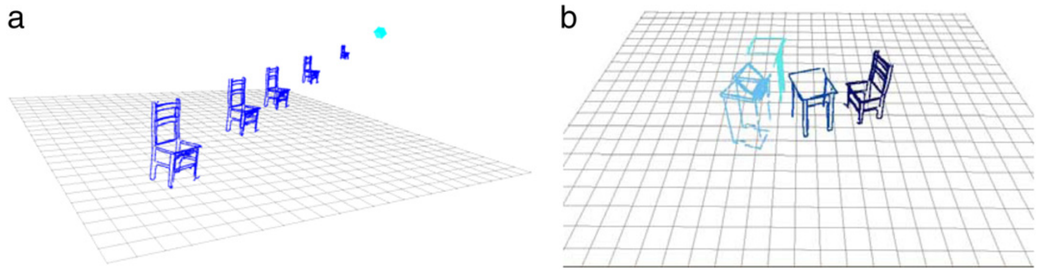

Fig. 7. (a) Recovering the size and position of a 3D object. The small green cube represents the 3D position of the robot's camera. (b) The image of a recovered 3D scene (for on-line demo go to: http://web.ics.purdue.edu/~li135/SceneRecover.html). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

judgments of relative distances and angles are very accurate when they are made with structured scenes like those in Fig. 1 (Kwon, 2012), but they are very inaccurate when the viewing conditions and stimuli are impoverished. For example, binocular observers make large systematic errors in judging distances when only a few points of light are shown in total darkness (Foley, 1972), and they misjudge angles formed by spherical objects suspended on strings from the ceiling (Doumen, Kappers, & Koenderink, 2006, 2007, 2008).

Fig. 7(b) shows the recovered 3D scene for the picture in Fig. 5. The online demo at http://web.ics.purdue.edu/~li135/SceneRecover.html shows an animation of this recovery. The widths and heights of the children's chairs were about 30 cm. The accuracy of our size and distance recovery can be evaluated by comparing their distances and sizes to the 20 cm. wide unit-square scale shown on the floor. The position and the orientation of the robot's camera used to make this image are indicated by the cyan box. These results show that our algorithm not only recovered the size and distance of the 3D objects accurately, it recovered the entire objects, including their invisible back contours!

We provided the reader with a demo to illustrate the methods described in this section. This demo consists of a Matlab program and data (http://web.ics.purdue.edu/~li135/JMP2011/JMPDemo.rar). The image shown in Fig. 5 is the image used for the 3D recovery in this demo. The 2D contours extracted from this image are provided as the input data. The Matlab program performs the 3D recovery of the contours as described in this section. The reader is encouraged to use this program to recover the 3D scene represented in Fig. 5, and also to use this program to recover contours in their own images after calibrating their own camera (see above).

3. Figure-Ground Organization (FGO)

In the Introduction, we enumerated the tasks that had to be accomplished when we want to recover a 3D scene. One of these tasks, isolating objects from their background, was called the Figure-Ground Organization (FGO) problem. The fundamental importance of this problem was pointed out by the Gestalt Psychologists almost 100 years ago, but they made very little progress in developing it primarily because they lacked the mathematical and computational tools to do so. Following the Cognitive Revolution, such tools became available: computers were built, Information Theory was formulated, and Cybernetics was established as an interdisciplinary specialty to integrate engineering, biology and psychology. Unfortunately, the progress made in the development of applied mathematics, computer science and electrical engineering did not include any important advances in our understanding of the most important basic problem in vision, viz., the FGO problem that our machine had to solve. This absence of significant progress with the FGO problem allowed the vision community to stop worrying about how important it was and, in time, they even began to denigrate this as well as many other contributions of the Gestalt Psychologists to visual perception. The few who did try to work on it tried to formulate theories and models of FGO without clarifying the ill-defined concepts used by Gestalt Psychologists before the Cognitive Revolution. This led to a lot of confusion in the machine vision community on one side and in the human vision community, on the other. This confusion would (and should) have been avoided if "visionists" on both sides had remembered the question that actually underlay the FGO problem, namely, how does the human observer see real 3D objects in natural 3D scenes veridically on the basis of the information provided by 2D real retinal images. Human beings, as well as other animals, obviously do. How do they do it? Ignoring, or downplaying the importance of studying real viewing conditions inevitably changed the nature of the problem. Confining efforts to the study of 2D stimuli should not, and did not, lead anywhere.

Finding objects in the 2D image and in the 3D scene

We begin by considering how the traditional approach tries to distinguish objects from their backgrounds. This approach uses information only present in a pair of 2D images with slightly different views of the scene. The pair can be obtained either

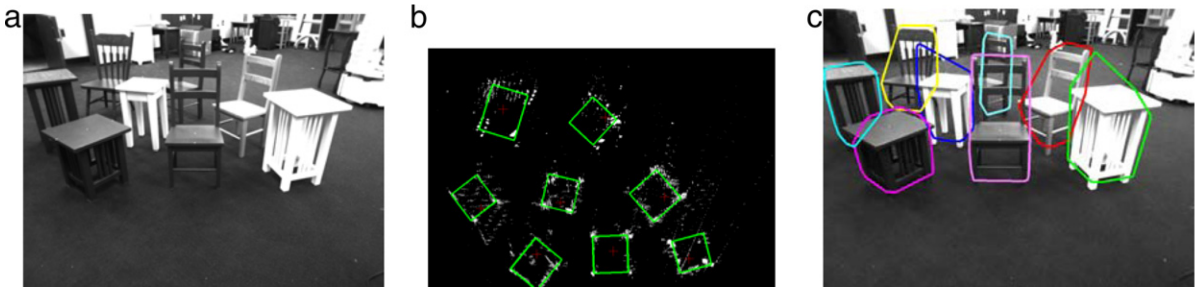

Fig. 8. (a) A 2D image of a 3D scene containing children's furniture. (b) A top view of the 3D scene in (a) showing the 8 objects that were "seen" by the robot (the robot analyzed the 3D scene within a 3 m viewing distance). The green rectangles represent individual objects, their sizes, aspect ratios and orientations. Note that even the occluded chair in the back of the scene was detected. The top view was produced from a pair of images acquired by the robot's stereoscopic camera. (c) The detected regions for individual objects in the 2D image. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

by using two eyes ("binocular disparity") or by using successive images from a single eye ("motion parallax"). Julesz (1971) provided strong support for the functional advantages inherent in having more than one view of a scene by showing that binocular disparity and motion parallax are critical in breaking camouflage. His most compelling support for this claim was obtained when he showed that perceptions of 3D spatial relations can be produced with "random-dot-stereograms". Such stereograms contain no useful monocular information about the objects that are actually present in the visual field.

Fig. 8 illustrates what can be accomplished by using two different images. The camera mounted on our robot acquired a pair of stereoscopic images that the robot used to detect and locate the 3D objects represented in its pair of 2D images (see Fig. 1). The robot started the process by using binocular disparity to compute a 3D map. This was done by using an off-the-shelf algorithm for solving the stereo-correspondence problem (Wong, Vassiliadis, & Cotofana, 2002). Two computational steps were then used by the robot to construct a top view of the 3D scene, specifically, the 3D points cloud was computed from stereo disparity by using the triangulation method, and the floor was approximated by finding a 3D plane which contained the maximal number of points.[6] Note that this step was also used to calibrate our robot's camera, namely, we computed the orientation of the robot's camera and its position relative to the floor (see the parameters of the extrinsic matrix). Since camera orientation is computed relative to the floor, the slant of the floor is not an issue. The detection of the floor is an important step because:

(1) one can remove the points close to the floor and beneath it once we know where it is;
(2) one can project the remaining 3D points onto the floor to generate a top view image. The white dots in Fig. 8(b) show the top view image after the floor points were removed. This made the layout of the objects in the scene very clear.

We then identified the number of objects, their positions and their orientations by fitting rectangles within the top view image. The 3D distances, sizes and aspect ratios may not be very accurate in this top view, but all 8 objects present in this scene were detected and located relative to each other quite well. This is evident in the 2D image of Fig. 8(b). It is clear that even at this early stage of analysis, the robot has obtained considerable information about this 3D scene. The top view clearly has sufficient information for the robot to plan navigations among all of these 8 objects. The top view of the furniture arrangement shown in Fig. 8(b) also makes it clear that the critical first step of FGO has been solved for this rather complex furniture arrangement. The computation of the top view and its use in solving the FGO problem makes intuitive sense because a top view of objects in most natural scenes is not likely to have one object occluding another. Furniture stacked in a storeroom might be a relatively common exception. Occlusions are common in the original view shown in Fig. 8(a) because far objects are likely to be occluded by near objects but this does not present a problem when the top view is used. Using a scene-centered, rather than a viewer-centered, representation early in processing proved to be essential for solving the FGO problem.

If you want to do more than navigate in this environment, it probably will become useful to recover the actual 3D shapes of each of the 8 objects. Their 3D shapes will be the best way to identify them because their shapes will let you know their purpose, sitting on some and eating off others. Doing this requires obtaining 2D information about the edges representing each 3D shape. Detecting meaningful edges in a single 2D image is difficult because there are always many spurious edges in the image caused by texture and shading. The problem can be solved if the region in the image representing each individual object can be specified. Fig. 8(c) illustrates how our model solved this problem. The model estimates the height of each object from the distribution of the 3D points that projected to a given rectangle in Fig. 8(b). This operation produced a 3D "bounding box" for each object. This 3D box is then projected to the original 2D image. This produces a convex region containing the image of the object. So, our method, as described in this section, can actually be used to produce both 3D and 2D FGO: it can also be used to determine the spatial location of each of the objects on the floor as well as in the 2D image. Note that the 2-dimensional FGO is based on a 3-dimensional FGO. The 3D FGO is easier to perform so it is not surprising that it is best done before the 2D FGO. The 2D FGO is also critical, however, because it provides a means to transition from both the texture and surface information that were used to produce the depth map to the contours that are essential for recovering individual 3D shapes.

Finally, note that binocular information about the objects in the scene can actually be used to verify and update the orientation of the camera in the scene coordinate system (see extrinsic camera calibration). Furthermore, a comparison of the 3D scene after and before the robot's movement can be used by the robot for self-localization and visual verification of its navigation.

4. Extracting relevant edges

Now that we know that the recovery of 3D shapes depends entirely on contours, and that texture and surfaces play no role, we can ask how can we extract relevant (hence meaningful) contours of objects within a given scene? We need to know which contours belong to which object. Furthermore, we need to know how to organize the 2D contours in the retinal image so that this

[6] In the application, the thickness of the floor plane is set to 10 mm. Therefore, our algorithm can deal with a floor that is not perfectly flat.

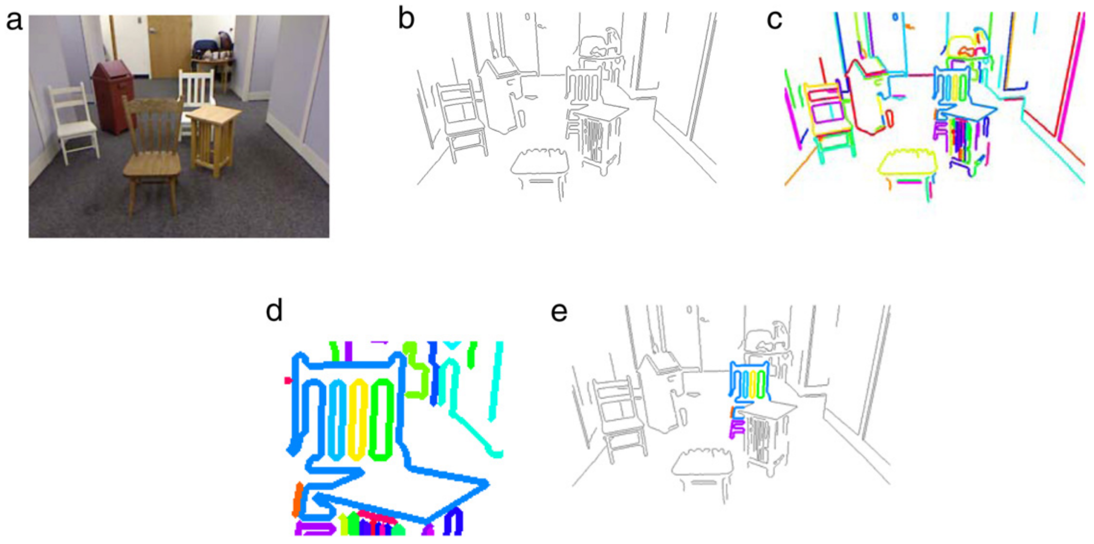

Fig. 9. (a) An input image. (b) Its binary edge image. (c) These edges grouped to edge fragments. (d) Occluding contours of a foreground object incorrectly merged with the background objects. (e) Contours of a single object selected in the image when the FGO problem has been solved correctly.

organization conveys sufficient information about the 3D shapes "out there" to permit an observer (human or robot) to function effectively in its environment. Considerable progress has been made in this direction recently. It is described in this and the next section.

The problem of extracting relevant (hence meaningful) edges in a 2D image that contains unfamiliar objects has traditionally been deemed to be an insoluble problem; a problem so difficult that many students of human and computer vision assume that a solution is actually impossible. The main difficulty traditionally seen arises from the fact that any edge-detection algorithm will detect at least 10 times as many edges as it should detect and will also miss some very important edges. These irrelevant (meaningless) edges are produced in the image by such things as texture, occlusions, shadows and by specular reflections.

In computer vision, where most of the work has been done so far, the process of extracting relevant edges is called "contour grouping". It begins by taking a color or a gray-level image (Fig. 9(a)) and reducing it to a binary image composed of edge pixels (Fig. 9(b)). The edge pixels are then grouped to form curves called "edge fragments". Different edge fragments are shown in different colors in Fig. 9(c). This is a greedy, low-level process. It uses only very simple processes such as proximity and good continuation and it uses them at the pixel level. Some other simple rules are often applied to eliminate edge fragments, e.g., the removal of too short or wiggly fragments. These processes, however, are not likely to extract the true contours of 3D objects. An example of this kind of failure is shown in Fig. 9(d) where contours of two different objects have been merged into a single contour. If such mistakes are not corrected, it will be impossible to perform a meaningful recovery of the 3D shape. The only algorithms developed to date that can produce correct results require familiarity with the 3D shape and its 2D images *before* the recovery can actually be made (Andriluka, Roth, & Schiele, 2008; Ferrari, Tuytelaars, & Gool, 2006; Latecki, Lu, Sobel, & Bai, 2008; Lin, Hua, & Davis, 2009; Ma & Latecki, 2011; Srinivasan, Zhu, & Shi, 2010; Toshev, Taskar, & Daniilidis, 2010; Yang & Latecki, 2010). These algorithms assume that the observer saw multiple views of each object and stored them in memory. These stored views are then used to match the edges in the retinal or camera image. Clearly, this multiple-view theory is not only very cumbersome; it is actually implausible because the contours of the 2D image change in unpredictable ways whenever the viewing direction changes. To actually use this approach, one would need a very large number of 2D models for each 3D object. Considering that there is a huge number of possible objects in our environment, a large number of possible positions, as well as the large range of sizes of the objects in the 2D image, the matching problem inherent in this kind of algorithm leads to a combinatorial explosion.

We already knew that this problem can be solved easily without familiarity by human beings who see unfamiliar shapes veridically. Our robot can do it without familiarity, too. Furthermore, it can do it easily once the specific 2D region that contains the image of a specific 3D object has been determined (see Fig. 8(c)). With this known, the algorithm focuses its analysis on the small set of edge fragments that are located within this region. Concentrating the analysis on a meaningful, predefined 2D region substantially improves the likelihood of extracting the relevant set of contours because contours belonging to the background have been eliminated from consideration (see Fig. 9(e)).

Once the relevant 2D contours have been extracted for each of the 3D objects, two types of 3D representations can be produced, namely, a coarse representation of the 3D shape in the form of a rectangular bounding box (Fig. 10(b)), and a more precise (finer) representation in the form of the 3D contours that represent each 3D shape (Fig. 10(c)). How are the two representations computed? Consider the bounding box first. It can be computed by using three vanishing points (see Fig. 11). One vanishing point is the intersection of symmetry lines (see Fig. 6). The second vanishing point is the intersection of the lines in the 2D image that are projections of the vertical lines in the 3D scene. Note that in the presence of gravity, very many natural objects, such as cats, dogs, birds, and human beings, as well as furniture have appendages with multiple vertical edges we call their "legs". Adjusting the orientation of these appendages permits them to maintain their balance when they stand or walk on tilted surfaces. This second vanishing point (which will be called here the "vertical vanishing point") can be determined solely on the basis of information about the direction of gravity, information that is readily available to living creatures (see Fig. 5). The vertical vanishing point is calculated as follows. Assume that the direction of gravity is N_g, then in the image the vanishing point for those 3D vertical lines can be expressed as:

$$v_g^* = MN_g. \tag{12}$$

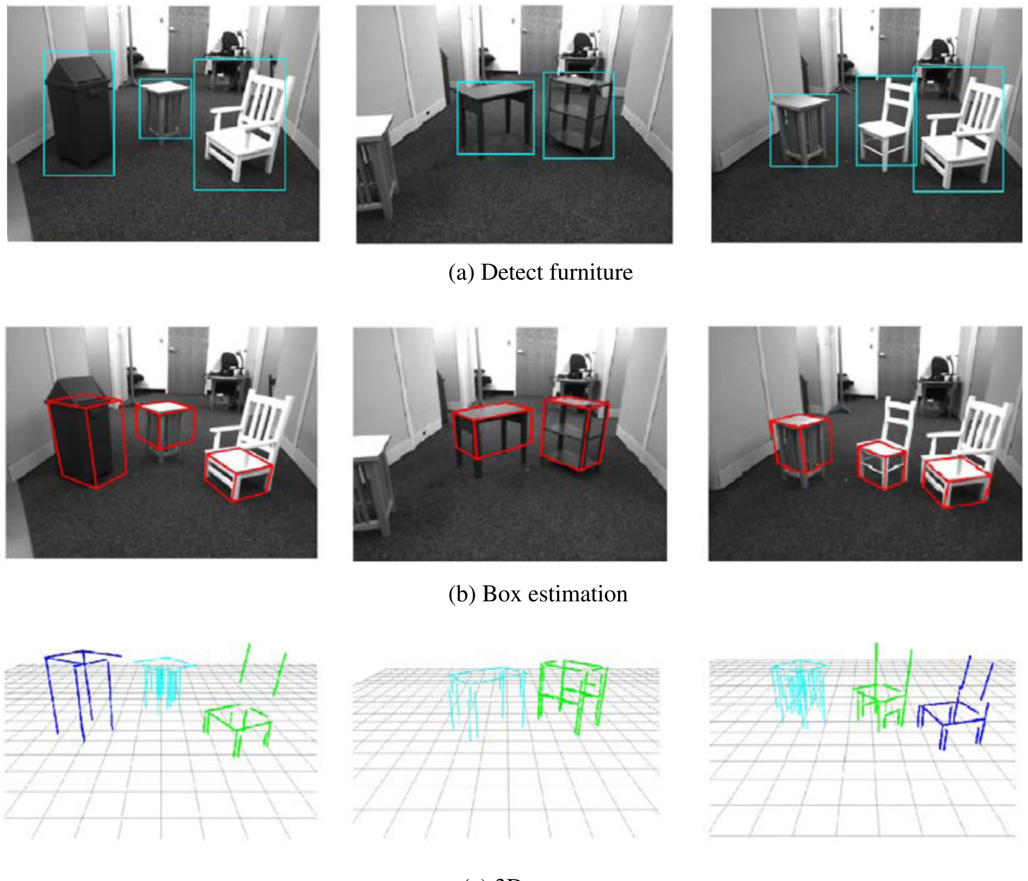

Fig. 10. (a) Detected objects. (b) Their 3D bounding boxes. (c) The recovered 3D shapes and locations.

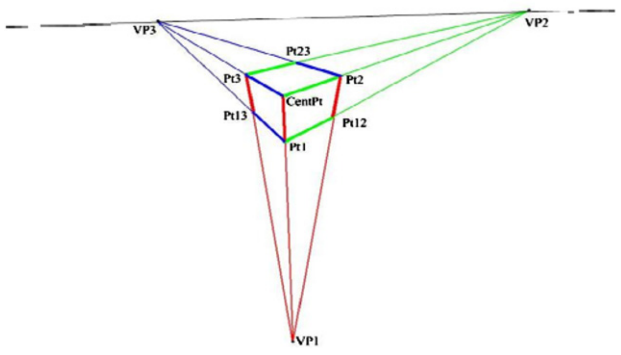

Fig. 11. A perspective projection of a box. VP_1, VP_2 and VP_3 represent the vanishing points for the three groups of mutually orthogonal line segments of the box.

Eq. (12) suggests that the vanishing point, like a horizon, is determined by the camera's intrinsic properties and the camera's orientation. Finally, note that for many objects, such as furniture, there is a third vanishing point that represents the edges orthogonal to the other two types of edges in 3D (Fig. 11). All three of these vanishing points form right angles with the vertex at the center of the perspective projection of the camera. This fact is equivalent to the following equation characterizing image properties (Eq. (8.7) in Hartley & Zisserman, 2003):

$$v_i^{*T} K^{-T} K^{-1} v_j^* = 0 \quad i \neq j. \tag{13}$$

The equation (just above) implies that if we know any two vanishing points, we can compute the third. In the case of animal bodies, which are not rectangular like chairs, the third vanishing point is also meaningful: it represents the direction in which the animal is facing. This means that a 3D rectangular bounding box computed on the basis of these 3 vanishing points is at least an adequate, albeit coarse representation for most objects, animate and inanimate.

Note that these 3D bounding boxes provide a type of information that is analogous to the way the boxes were estimated when we solved the binocular FGO problem (see Fig. 8(b)). The difference is that the boxes in Fig. 8(b) were estimated on the basis of texture information, whereas the boxes in Fig. 10(b) were estimated on the basis of contour information. One might consider the fact that these two different analyses led to the same result as an instantiation of the action of Grossberg's feature and boundary contour systems (Grossberg & Mingolla, 1985). There is an important difference, namely, our analyses are taking place at the stage of the 3D representation, the stage at which the 3D objects actually reside.

A precise (fine) representation of the 3D shape can be produced by performing the 3D shape recovery on the basis of the object's symmetry (see Fig. 10(c)). This recovery is done by using the algorithm described in Section 2. The only operation that remains to be done is the detection of the 3D mirror symmetry in the 2D asymmetrical image. This operation is explained in Section 5.

5. Establishing the 3D symmetry correspondence of contours

A 2D image of a 3D mirror-symmetrical object is itself symmetrical but only for a narrow range of 3D viewing directions, so the question arises as to how a 3D symmetry can be detected in a 2D asymmetrical image. This problem is not trivial because pairs of unrelated 2D curves always have a 3D symmetrical interpretation (Sawada, Li, & Pizlo, 2011). In other words, without additional constraints, 3D symmetry is accidental. This fact is illustrated

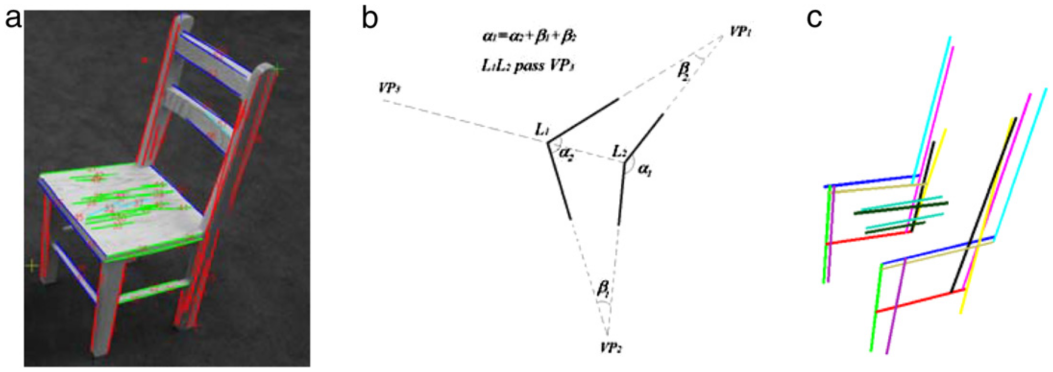

Fig. 12. (a) Extracted curves for the image of a chair. Curves corresponding to the same vanishing points are drawn in the same color. (b) Two necessary conditions that are used to check whether the two edges of a junction are the potential symmetrical edges of another junction. (c) Detected symmetrical curves for the image in (a). Symmetrical curves are drawn in the same color. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

in http://www1.psych.purdue.edu/~zpizlo/sym2011/DemoFiles/Demo8.html where two different symmetrical interpretations of a 2D curve are given. One of these interpretations is natural in the sense that it agrees with the percept of an observer produced by a stationary 2D curve. The other interpretation is surprising. The difference between these two interpretations is that the natural interpretation consists of two planar curves. The fact that the human visual system uses a planarity constraint has been known at least since Leclerc and Fischler (1992) and Sinha and Adelson (1992) published their models, but it was less clear why a planarity constraint is actually used. Planar contours are quite common in man-made objects, but they are much less common in biological organisms. We believe that the human visual system uses the planarity of contours constraint because this constraint eliminates spurious symmetrical 3D interpretations rather than because planar contours are common. It is important to point out that the use of the planarity constraint does not imply that the 3D interpretations have planar contours; it only implies that the interpretations have contours that are biased towards planarity. This means that the torsion of 3D curves is kept to a minimum. It turns out that minimizing torsion eliminates 3D interpretations that correspond to degenerate views, views that preclude the veridical perception of 3D objects (Sawada et al., 2011).

If two 3D curves are planar and mirror symmetrical, their 2D images are related by a 2D affine transformation in the case of an orthographic image and by a 2D projective transformation in the case of a perspective image. The fact that a 3D symmetry can be detected in a 2D asymmetrical image through an application of a 2D transformation and its invariants simplifies the problem substantially. Under such conditions, 3D symmetry becomes non-accidental in the sense that a 2D image of a 3D asymmetrical shape is unlikely to have 3D symmetrical interpretations.

The task of detecting 3D symmetry in a 2D image is always simplified if higher order features, such as corners, intersections, complex curves and closed contours are detected first. Two pairs of feature points in a 2D image correspond to two pairs of mirror symmetrical points in a 3D interpretation only if the line segments connecting the 2D corresponding points are parallel in an orthographic image and if they intersect at a vanishing point in a perspective image. Recall from the section on 3D scene recovery that vanishing points and lines can be estimated directly from vestibular cues provided by gravity. The pictures at http://web.ics.purdue.edu/~li135/SymDetect.html show the result of establishing 3D symmetry correspondence for a few objects shown in 2D perspective images. In this example, our model checked whether any two junctions in the 2D images satisfy the following two constraints: (1) the line connecting the junctions passes through one vanishing point and (2) the difference between two junction angles is equal to the sum of the angles formed by the two junctions with the other two vanishing points (see Fig. 12(b)). These are the necessary conditions for a 3D symmetrical interpretation in the case where the pair of symmetrical curves are on parallel planes (like in the case of two sides of a chair shown in Fig. 12(a)). The symmetrical pairs of curves are drawn in the same color. Once the contours have been organized, it is relatively easy to recover the 3D scene including the 3D shapes contained within it.

Summary and conclusion

We described a set of computational tools (models) that allow a robot to "see" a natural 3D scene and to "understand" it in the sense that it can recover the 3D shapes, sizes and locations of the objects in the scene as well as the free spaces among them. The Figure-Ground Organization and 3D shape recovery tools built into our robot permit it to perform both of these complicated tasks on its own. Our model does not use learning and it need not memorize hundreds of 2D images of 3D objects, so it can be used to recover unfamiliar scenes and unknown objects.

Next, we will briefly describe several prior attempts to recover shapes and scenes, relevant to our work, that are not discussed in the main text of our paper. This description is not intended to be a comprehensive review of prior work. Only highlights of the prior work, closely related to our model, will be included.

Our model is not the first attempt to apply simplicity constraints to recover 3D shapes and scenes. For example, Attneave and Frost (1969) used the orthogonality constraint to recover a 3D rectangular box from a single 2D image. Leclerc and Fischler (1992), Marill (1991), Sinha and Adelson (1992) and Sinha (1995) used minimum variance of angles, planarity, and minimum depth range to recover polyhedra from a single 2D image. All of these models were only tested with simple line drawings and were never applied to real images of real objects or real scenes.

Vanishing points in a 2D image can serve as an extension of the use of symmetry to recover 3D shapes. Vanishing points have been used in reconstructing 3D scenes, especially for urban (carpentered) environments. In such environments, parallel lines are fairly common and they are often vertical or horizontal. This is true about the sides of a road and the edges of typical rectangular rooms. With a calibrated camera, the 3D orientation of parallel lines can be derived from the vanishing point (Coxeter, 1987). This makes it possible to estimate the orientation of the camera relative to the 3D scene. A typical application of the vanishing point can be found in robot navigation (see DeSouza & Kak, 2002, for a survey of robot navigation).

In our model, we use binocular information to detect the floor and then isolate different objects (FGO) on the floor. Burschka, Lee, and Hager (2002) used a similar method to detect objects from a disparity image, allowing their robot to avoid obstacles during its navigation. Their algorithm, however, relied on the spatial continuity of binocular disparity, which meant that the robot could not detect partially-occluded objects that are divided into several regions in the image.

Despite considerable progress, there is still a lot of work to be done. By far the most important unfinished business is to implement the algorithms in such a way that the robot can see the environment in "real time". In human vision there are always delays due to neural transmission from the retina to the visual cortex and due to visual processing itself (Nijhawan, 2002). The robot has to deal with the same kind of delays. Acquiring and analyzing images takes time, so when the 3D reconstruction is finished, the 3D representation corresponds to the scene that was present 100 ms, or more, ago. If the scene is static, this temporal delay is not a problem. But when the scene is dynamic because of motion of the seeing agent or the motion of other objects, the only way to actually see the world "now" (in real time) is to make temporal predictions. Temporal predictions can be accurate for the same reason that spatial predictions can be accurate. The fact that our model can recover the back invisible parts of opaque objects, and the invisible spaces behind objects shows that spatial predictions can be quite pervasive. Both temporal prediction and spatial prediction work very well because they are based on very effective *a priori* constraints. *Put differently, the only way to actually perform in real time, as opposed to performing with respect to what happened* 100 ms *in the past is to see our predictions of the world, not the world itself.*

These problems are closely related to problems explored in active vision approaches. Active vision is a well established research area in computer vision (e.g., Blake & Yuille, 1992). An active vision system is a dynamic system that is able to change the location and orientation of the cameras in order to investigate the environment and get better information from it. Active Vision favors a goal oriented exploration of the environment in order to answer specific queries posed by the observer. It is based on the premise that an observer (human or computer) may be able to understand a visual environment more effectively and efficiently if the viewpoint and viewing direction of the camera are dynamic and task dependent.

Last, but by no means the least task facing us, the models currently used for extracting all relevant contours of 3D shapes should be elaborated so that the robot can extract 3D shapes and scenes with the high degree of precision characteristic of the human visual system.

Note that all of these unfinished projects are actually elaborations of the 3D shape and scene recovery models that we have in hand now. In other words, all of the major steps required for 3D shape and scene recovery have been accomplished and they can be performed autonomously by a robot at this time. The remaining steps (described just above) are designed to enhance its performance, bringing it in line with the performance of human beings. Once they are accomplished, our robot will not only be able to act autonomously; it will also be able to navigate within natural scenes as well as a human being can navigate under similar conditions. There is even good reason to believe that the FGO and 3D shape-recovery tools that you have seen work so well for our robot are actually rather similar to those used by human beings performing similar tasks. We feel entitled to make this claim because when we tested these tools in human psychophysical experiments, the human and the model's performances were very similar (Kwon, 2012; Li et al., 2011). Even if we ignore this psychophysical support, the mere fact that the robot can actually "see" a 3D scene veridically and plan its actions effectively within it provides evidence for our belief that the robot's tools are at least biologically-plausible even if they ultimately prove to be different from those actually used in the human visual system. It is important in evaluating these provocative claims to keep in mind that no other existing machine vision system has even come close to approximating the performance of the human being in even very simple visual environments. Our machine vision system approximates human performance very well in relatively complex, naturalistic environments. Discovering that a system like ours can do this provided us with some new insights into how a visual system like ours accomplishes what it does so well. Finally, explaining how the human visual system works has been only one of the goals of our work.

It has not gone unnoticed that robots, equipped with the novel kind of computational visual system described in this tutorial, will be able to deliver food and supplies in hospitals and trim grass on lawns at least as well as conventional contemporary robots can perform such tasks, but conventional robots, unlike ours, accomplish these tasks by using tools that do not resemble those used by humans beings. Our robot's tools do and this fundamental difference opens up the possibility of having a machine emulate a wide range of human activities within quite complex natural environments. This becomes possible because our machine and human beings perform effective navigations without measuring absolute distances. Both navigate by constructing a limited number of accurate representations of 3D shapes. All other contemporary robots base their navigations on making many iterative measurements of absolute distances. The simple visual/gravitational method used by our machine for FGO, 3D shape and 3D scene recovery probably works so well because it emulates the method that human beings, and many other successful animals, honed during the millennia required for their evolution.

Acknowledgments

This project was supported by the NSF, AFOSR and DoD grants.

Appendix A. The recovery of a pair of symmetrical points

Suppose x is a point in a 2D image and it is expressed by Euclidean coordinates. Its corresponding homogenous coordinates can be written as

$$x^* = \begin{pmatrix} x^T & 1 \end{pmatrix}^T . \tag{A.1}$$

Suppose M is a 3×3 matrix that consists of the first three column vectors of a camera matrix and p_4 is the fourth column vector of the camera matrix. Then all 3D points whose image is x is expressed as follows (Hartley & Zisserman, 2003)

$$X = M^{-1}(kx^* - p_4) \tag{A.2}$$

in which k is a free parameter. Note that X is a vector with the Euclidean coordinates of a 3D point. Suppose X_1 and X_2 are a pair of symmetric 3D points, and their images are x_1 and x_2. Then from Eq. (A.2), we obtain

$$X_i = M^{-1}(k_i x_i^* - p_4) \quad i = 1, 2. \tag{A.3}$$

Therefore, to recover the 3D point X_1 and X_2 from their images x_1 and x_2, we need to compute k_1 and k_2 in Eq. (A.3). Suppose the direction of the line X_1X_2 is V, then V and the vanishing point of symmetry lines v^* must satisfy the following equation

$$V = M^{-1}v^*. \tag{A.4}$$

Note that V represents not only the direction of X_1X_2, but also the normal of symmetry plane. Therefore, the symmetry plane for X_1 and X_2 can be expressed as

$$\pi = \begin{pmatrix} V^T & d \end{pmatrix}^T \tag{A.5}$$

where d indicates the position of the symmetry plane.

X_1 and X_2 are symmetric with respect to the symmetry plane π if and only if the following two conditions are satisfied

(1) the line X_1X_2 is perpendicular to the symmetry plane π;
(2) the midpoint of X_1 and X_2 is on the symmetry plane π.

From condition (1), we can derive

$$X_1 - X_2 = k_v V \tag{A.6}$$

where k_v is a free parameter. Combining Eq. (A.6) with Eqs. (A.3) and (A.4), we obtain

$$M^{-1}(k_1x_1^* - p_4) - M^{-1}(k_2x_2^* - p_4) = k_vM^{-1}v^*. \tag{A.7}$$

Left multiplying by M both sides of Eq. (A.2), we obtain

$$k_1x_1^* - k_2x_2^* = k_vv^*. \tag{A.8}$$

Replacing the homogenous coordinates for x_1, x_2 and v in Eq. (A.8) with their Euclidean coordinates, we obtain

$$k_1\begin{pmatrix}x_1\\1\end{pmatrix} - k_2\begin{pmatrix}x_2\\1\end{pmatrix} = k_v\begin{pmatrix}v\\1\end{pmatrix}. \tag{A.9}$$

Eq. (A.9) could be decomposed to two equations

$$k_1 - k_2 = k_v \tag{A.10}$$

and

$$k_1x_1 - k_2x_2 = k_vv. \tag{A.11}$$

Replacing k_v in Eq. (A.11) with the expression in Eq. (A.9), we obtain

$$k_1(x_1 - v) = k_2(x_2 - v). \tag{A.12}$$

Eq. (A.11) implies two facts

(a) x_1, x_2 and v are collinear. Thus, Eq. (A.12), which represents two linear equations, is redundant.
(b) the ratio between k_1 and k_2 is proportional to the ratio of the vectors x_2v and x_1v.

x_1 and x_2 are on the same side of v. Otherwise, one of the recovered points X_1 and X_2 is behind the image plane. Therefore, Eq. (A.11) can be written as

$$k_1\,|x_1 - v| = k_2\,|x_2 - v| \tag{A.13}$$

where $|x_1 - v|$ and $|x_2 - v|$ represent the distance from x_1 and x_2 to v. Eq. (A.13) suggests that the ratio of k_1 and k_2 is proportional to the ratio of the distance from two symmetric points to the vanishing point.

From condition (2), we can derive

$$\frac{X_1 + X_2}{2}V + d = 0. \tag{A.14}$$

Combining Eq. (A.14) with Eqs. (A.3) and (A.4), we obtain

$$\begin{pmatrix}v^{*T}M^{-T}M^{-1}x_1^* & v^{*T}M^{-T}M^{-1}x_2^*\end{pmatrix}\begin{pmatrix}k_1\\k_2\end{pmatrix} = 2v^{*T}M^{-T}M^{-1}p_4 - 2d. \tag{A.15}$$

Combining Eqs. (A.13) and (A.15), we obtain

$$\begin{pmatrix}v^{*T}M^{-T}M^{-1}x_1^* & v^{*T}M^{-T}M^{-1}x_2^*\\ |x_1 - v| & -|x_2 - v|\end{pmatrix}\begin{pmatrix}k_1\\k_2\end{pmatrix} = \begin{pmatrix}2v^{*T}M^{-T}M^{-1}p_4 - 2d\\ \vec{0}_{2x1}\end{pmatrix}. \tag{A.16}$$

Appendix B. Estimating the vanishing point on a horizon from a set of symmetry lines

Let the horizon be expressed as

$$l^* = \begin{pmatrix}l_x & l_y & l_w\end{pmatrix}^T \tag{B.1}$$

then $v_0^* = \begin{pmatrix}-l_y & l_x & 0\end{pmatrix}^T$ and $v_1^* = \begin{pmatrix}0 & -l_w & l_y\end{pmatrix}^T$[7] are two points on the horizon because $l^{*T}v_0^* = 0$ and $l^{*T}v_1^* = 0$. Specifically v_0^* is a point at infinity on the horizon. Then all points on l^* can be expressed as

$$v^* = v_0^* + \lambda v_1^*. \tag{B.2}$$

Let $l_1^*, l_2^*, \ldots, l_k^*$ represent the k symmetry lines. Ideally, if all symmetric lines intersect with the horizon at point v^*, then the following equations will be satisfied

$$l_i^{*T}v^* = 0 \quad i = 1, 2, \ldots, k. \tag{B.3}$$

Let $B = \begin{pmatrix}l_1^{*T} & l_2^{*T} & \cdots & l_k^{*T}\end{pmatrix}^T$, then Eq. (B.3) is expressed as

$$Bv^* = \vec{0}_{kx1}. \tag{B.4}$$

Combining Eqs. (B.2) and (B.4), we obtain

$$\lambda Bv_1^* = -Bv_0^*. \tag{B.5}$$

Therefore, the problem of estimating the vanishing point on the horizon is changed to finding the optimum value of λ that satisfies Eq. (B.5) in the least square sense. It follows

$$\lambda = ((Bv_1^*)^T(Bv_1^*))^{-1}(Bv_1^*)^T(-Bv_0^*) \tag{B.6}$$

or

$$\lambda = -\frac{v_1^{*T}B^TBv_0^*}{v_1^{*T}B^TBv_1^*}. \tag{B.7}$$

Appendix C. Pseduo code for recovering a 3D scene from a single 2D image

```
Objects recover3DScene(Camera camera, Image image)
  {
      set objects empty;
      // compute the horizon line
      horizon = camera.calHorizon();
      // compute the floor plane
      floor = camera.calFloor();
      for(i=0;i <image.objnum;++i)
      {
          // estimate the vanishing point on the horizon for object i
          vanishingPnt = estVanishingPnts(image.symlines, horizon);
          // determine the symmetric correspondences on symmetric curves
          symcorrespond = detSymCorrespond(vanishingPnt, image.symcurves);
          // Recover symmetric correspondences
          recsympnts = recSymPnts(camera, symcorrespond);
          // Recover hidden part
          rechiddenpnts = recHiddenPnts(camera, recsympnts, image.planarcurves);
          // Scale recovered objects
          objects[i] = scaleObjects(floor, recsympnts, rechiddenpnts);
      }
  }
```

[7] If l_y is 0, then v_0^* and v_1^* represent the same point. For this case, we can set $v_1^* = \begin{pmatrix}-l_w & 0 & l_x\end{pmatrix}^T$. If both l_x and l_y are 0, then l^* represents the line at infinity and it is the vanishing lines for the planes that are parallel to the image plane. For this case, we can set $v_0^* = \begin{pmatrix}1 & 0 & 0\end{pmatrix}^T$ and $v_1^* = \begin{pmatrix}0 & 1 & 0\end{pmatrix}^T$.

References

Andriluka, M., Roth, S., & Schiele, B. (2008). People-tracking-by-detection and people-detection-by-tracking. CVPR 2008.
Attneave, F., & Frost, R. (1969). The determination of perceived tridimensional orientation by minimum criteria. *Perception & Psychophysics*, 6, 391–396.

Bartlett, F. (1932). *Remembering*. Cambridge: Cambridge University Press.

Blake, A., & Yuille, A. (Eds.) (1992). *Active vision*. MIT Press.

Burschka, D., Lee, S., & Hager, G. (2002). Stereo-based obstacle avoidance in indoor environments with active sensor re-calibration. In *Proceedings of the IEEE international conference on robotics and automation*. ICRA2002. Washington, DC. May.

Coxeter, H. S. M. (1987). *Projective geometry* (2nd ed.). New York: Springer.

DeSouza, G. N., & Kak, A. C. (2002). Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*, 237–267.

Dewey, J. (1896). The reflex arc concept in psychology. *Psychological Review*, *3*, 357–370.

Doumen, M. J., Kappers, A. M. L., & Koenderink, J. J. (2006). Horizontal–vertical anisotropy in visual space. *Acta Psychologica*, *123*, 219–239.

Doumen, M. J., Kappers, A. M. L., & Koenderink, J. J. (2007). Effects of context on a visual 3-D pointing task. *Perception*, *36*, 75–90.

Doumen, M. J., Kappers, A. M. L., & Koenderink, J. J. (2008). Do reference surfaces influence exocentric pointing? *Acta Psychologica*, *128*, 310–317.

Faugeras, O. (2001). *Three-dimensional computer vision—a geometric viewpoint*. Cambridge, MA: MIT Press.

Ferrari, V., Tuytelaars, T., & Gool, L. V. (2006). Object detection with contour segment networks. ECCV, 2006.

Foley, J. M. (1972). The size-distance relation and intrinsic geometry of visual space: implications for processing. *Vision Research*, *12*, 323–332.

Gibson, J. J. (1966). *The senses considered as perceptual systems*. Boston: Houghton Mifflin.

Grossberg, S., & Mingolla, E. (1985). Neural dynamics of form perception: boundary completion, illusory figures, and neon color spreading. *Psychological Review*, *92*, 173–211.

Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge: Cambridge.

Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.

James, W. (1890). *The principles of psychology*. New York: Dover.

Julesz, B. (1971). *Foundations of cyclopean perception*. Chicago: University of Chicago Press.

Kubovy, M. (1986). *The psychology of perspective and renaissance art*. Cambridge: Cambridge University Press.

Kwon, T. (2012). Human recovery of the shape and size of a 3D indoor scene. Unpublished master's thesis. Department of psychological sciences. Purdue University.

Latecki, L. J., Lu, C., Sobel, M., & Bai, X. (2008). Multiscale random fields with application to contour grouping. In *Neural information processing systems conf.* NIPS, 2008.

Leclerc, Y. G., & Fischler, M. A. (1992). An optimization-based approach to the interpretation of single line drawings as 3D wire frames. *International Journal of Computer Vision*, *9*, 113–136.

Li, Y. (2009). Perception of parallelepipeds: Perkins' law. *Perception*, *38*, 1767–1781.

Li, Y., Pizlo, Z., & Steinman, M. R. (2009). A computational model that recovers the 3D shape of an object from a single 2D retinal representation. *Vision Research*, *49*, 979–991.

Li, Y., Sawada, T., Shi, Y., Kwon, T., & Pizlo, Z. (2011). A Bayesian model of binocular perception of 3D mirror symmetrical polyhedral. *Journal of Vision*, *11*(4), 11, 1–20 (http://www.journalofvision.org/content/11/4/11).

Lin, Z., Hua, G., & Davis, L. S. (2009). Multiple instance feature for robust part-based object detection. CVPR 2009.

Ma, T., & Latecki, L. J. (2011). From partial shape matching through local deformation to robust global shape similarity for object detection. In *IEEE conf. on computer vision and pattern recognition*, CVPR, June.

Marill, T. (1991). Emulating the human interpretation of line drawings as three-dimensional objects. *International Journal of Computer Vision*, *6*, 147–161.

Marr, D. (1982). *Vision*. San Francisco: W.H. Freeman.

Matin, L., & Fox, R. C. (1989). Visually perceived eye level and perceived elevation of objects: linearly additive influences from visual field pitch and from gravity. *Vision Research*, *29*(3), 315–324.

Miller, G. A., Galanter, E., & Pribram, K. H. (1960). *Plans and the structure of behavior*. New York: Holt.

Nijhawan, R. (2002). Neural delays, visual motion and the flash-lag effect. *Trends in Cognitive Sciences*, *6*(9), 387–393.

Pirenne, M. H. (1970). *Optics painting & photography*. London: Cambridge.

Pizlo, Z. (2001). Perception viewed as an inverse problem. *Vision Research*, *41*, 3145–3161.

Pizlo, Z. (2008). *3D shape: its unique place in visual perception*. Cambridge, MA: MIT Press.

Pizlo, Z., Sawada, T., Li, Y., Kropatsch, G. W., & Steinman, M. R. (2010). New approach to the perception of 3D shape based on veridicality, complexity, symmetry and volume. *Vision Research*, *50*, 1–11.

Poggio, T., Torre, V., & Koch, C. (1985). Computational vision and regularization theory. *Nature*, *317*, 314–319.

Sawada, T. (2010). Visual detection of symmetry in 3D shapes. *Journal of Vision*, *10*(6), 4, 1–22 (http://www.journalofvision.org/content/10/6/4).

Sawada, T., Li, Y., & Pizlo, Z. (2011). Any pair of 2D curves is consistent with a 3D symmetric interpretation. *Symmetry*, *3*, 365–388.

Sawada, T., & Pizlo, Z. (2008). Detection of skewed symmetry. *Journal of Vision*, *8*(5), 14, 1–18 (http://www.journalofvision.org/content/8/5/14).

Sinha, P. (1995). Perceiving and recognizing three-dimensional forms. Massachusetts Institute of Technology. ProQuest Dissertations and Theses, 1 (http://search.proquest.com/docview/304225734?accountid=13360).

Sinha, P., & Adelson, E. H. (1992). Recovery of 3-D shape from 2-D wireframe drawings. *Investigative Ophthalmology & Visual Science*, *33*(Suppl.), 825.

Srinivasan, P., Zhu, Q., & Shi, J. (2010). Many-to-one contour matching for describing and discriminating object shape. CVPR, 2010.

Steinman, R. M. (1965). Effect of target size, luminance and color on monocular fixation. *Journal of the Optical Society of America*, *55*, 1158–1165.

Toshev, A., Taskar, B., & Daniilidis, K. (2010). Object detection via boundary structure segmentation. In *IEEE comp. vision pattern recognition,* CVPR.

Wong, S., Vassiliadis, S., & Cotofana, S. D. (2002). A sum of absolute differences implementation in hardware. In *Proceeding of the 28th euromicro conference*. (pp. 183–188).

Yang, X., & Latecki, L. J. (2010). Weakly supervised shape based object detection with particle filter. In *European conference on computer vision,* ECCV.