Introduction to Autonomous Mobile Robots

Contents

feedback loops. In higher-end cameras, the user can modify the values of these parameters via software. The *iris position* and *shutter speed* regulate the amount of light being measured by the camera. The iris is simply a mechanical aperture that constricts incoming light, just as in standard 35 mm cameras. Shutter speed regulates the integration period of the chip. In higher-end cameras, the effective shutter speed can be as brief at 1/30,000 seconds and as long as 2 seconds. *Camera gain* controls the overall amplification of the analog signal, prior to A/D conversion. However, it is very important to understand that even though the image may appear brighter after setting high gain, the shutter speed and iris may not have changed at all. Thus gain merely amplifies the signal, and it amplifies along with the signal all of the associated noise and error.

The key disadvantages of CCD cameras are primarily in the areas of inconstancy and dynamic range. As mentioned earlier, a number of parameters can change the brightness and colors with which a camera creates its image. Manipulating these parameters in a way to provide consistency over time and over environments, for example, ensuring that a green shirt always looks green, and something dark gray is always dark gray, remains an open problem in the vision community. For more details on the fields of color constancy and luminosity constancy, consult [65].

The second class of disadvantages relates to the behavior of a CCD chip in environments with extreme illumination. In cases of very low illumination, each pixel will receive only a small number of photons. The longest possible integration period (i.e., shutter speed) and camera optics (i.e., pixel size, chip size, lens focal length and diameter) will determine the minimum level of light for which the signal is stronger than random error noise. In cases of very high illumination, a pixel fills its well with free electrons. As the well reaches its limit, the probability of trapping additional electrons falls, and therefore the linearity between incoming light and electrons in the well degrades. Termed *saturation*, this can indicate the existence of a further problem related to cross-sensitivity. When a well has reached its limit, then additional light within the remainder of the integration period may cause further charge to leak into neighboring pixels, causing them to report incorrect values or even reach secondary saturation. This effect, called *blooming*, means that individual pixel values are not truly independent.

The camera parameters may be adjusted for an environment with a particular light level, but the problem remains that the dynamic range of a camera is limited by the well capacity (also called *well depth*) of the individual pixels. The well depth typically ranges between 20,000 and 350,000 electrons. For example, a high-quality CCD may have pixels that can hold 40,000 electrons. The noise level for reading the well may be 11 electrons, and therefore the dynamic range will be 40,000:11, or 3600:1, which is 35 dB.

CMOS cameras. The complementary metal oxide semiconductor chip is a significant departure from the CCD. It, too, has an array of pixels, but located along the side of each

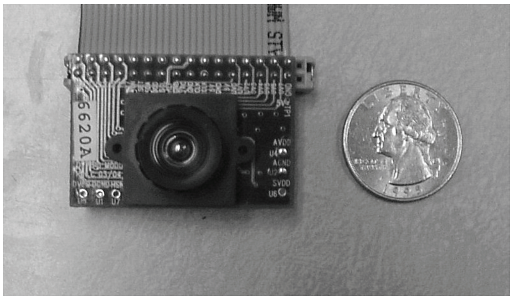


Figure 4.26
A commercially available, low-cost CMOS camera with lens attached.

pixel are several transistors specific to that pixel. As in CCD chips, all of the pixels accumulate charge during the integration period. During the data collection step, the CMOS takes a new approach: the pixel-specific circuitry next to every pixel measures and amplifies the pixel's signal, all in parallel for every pixel in the array. Using more traditional traces from general semiconductor chips, the resulting pixel values are all carried to their destinations.

CMOS has a number of advantages over CCD technologies. First and foremost, there is no need for the specialized clock drivers and circuitry required in the CCD to transfer each pixel's charge down all of the array columns and across all of its rows. This also means that specialized semiconductor manufacturing processes are not required to create CMOS chips. Therefore, the same production lines that create microchips can create inexpensive CMOS chips as well (see figure 4.26). The CMOS chip is so much simpler that it consumes significantly less power; incredibly, it operates with a power consumption that is one-hundredth the power consumption of a CCD chip. In a mobile robot, especially flying, power is a scarce resource and therefore this is an important advantage.

Traditionally, CCD sensors outperformed CMOS in quality sensitive applications such as digital single-lens-reflex cameras, while CMOS was better for low-power applications, but today, CMOS is used in most digital cameras.

Given this summary of the mechanism behind CCD and CMOS chips, one can appreciate the sensitivity of any vision robot sensor to its environment. As compared to the human eye, these chips all have far poorer adaptation, cross-sensitivity, and dynamic range. As a result, vision sensors today continue to be fragile. Only over time, as the underlying performance of imaging chips improves, will significantly more robust vision sensors for mobile robots be available.

Camera output considerations. Although digital cameras have inherently digital output, throughout the 1980s and early 1990s, most affordable vision modules provided analog output signals, such as NTSC (National Television Standards Committee) and PAL (Phase Alternating Line). These camera systems included a D/A converter which, ironically, would be counteracted on the computer using a *frame grabber*, effectively an A/D converter board situated, for example, on a computer's bus. The D/A and A/D steps are far from noise free, and furthermore the color depth of the analog signal in such cameras was optimized for human vision, not computer vision.

More recently, both CCD and CMOS technology vision systems provide digital signals that can be directly utilized by the roboticist. At the most basic level, an imaging chip provides parallel digital I/O (input/output) pins that communicate discrete pixel level values. Some vision modules make use of these direct digital signals, which must be handled subject to hard-time constraints governed by the imaging chip. To relieve the real-time demands, researchers often place an *image buffer chip* between the imager's digital output and the computer's digital inputs. Such chips, commonly used in webcams, capture a complete image snapshot and enable non-real-time access to the pixels, usually in a single, ordered pass.

At the highest level, a roboticist may choose instead to utilize a higher-level digital transport protocol to communicate with an imager. Most common are the IEEE 1394 (Firewire) standard and the USB (and USB 2.0) standards, although some older imaging modules also support serial (RS-232). To use any such high-level protocol, one must locate or create driver code both for that communication layer and for the particular implementation details of the imaging chip. Take note, however, of the distinction between lossless digital video and the standard digital video stream designed for human visual consumption. Most digital video cameras provide digital output, but often only in compressed form. For vision researchers, such compression must be avoided as it not only discards information but even introduces image detail that does not actually exist, such as MPEG (Moving Picture Experts Group) discretization boundaries.

Color camera. The basic light-measuring process described before is colorless: it is just measuring the total number of photons that strike each pixel in the integration period. There are two common approaches for creating *color* images, which use a single chip or three separate chips.

The single chip technology uses the so-called *Bayer* filter. The pixels on the chip are grouped into 2×2 sets of four, then red, green, and blue color filters are applied so that each individual pixel receives only light of one color. Normally, two pixels of each 2×2 block measure green while the remaining two pixels measure red and blue light intensity (figure 4.27). The reason there are twice as many green filters as red and blue is that the

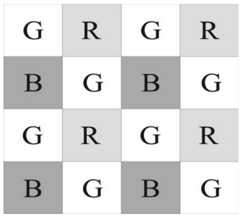


Figure 4.27 Bayer color filter array.

luminance signal is mostly determined by green values, and the visual system is much more sensitive to high frequency detail in luminance than in chrominance. The process of interpolating the missing color values so that we have valid RGB values as all the pixels is known as *demosaicing*. Of course, this one-chip technology has a geometric resolution disadvantage. The number of pixels in the system has been effectively cut by a factor of four, and therefore the image resolution output by the camera will be sacrificed.

The three-chip color camera avoids these problems by splitting the incoming light into three complete (lower intensity) copies. Three separate chips receive the light, with one red, green, or blue filter over each entire chip. Thus, in parallel, each chip measures light intensity for one color, and the camera must combine the chips' outputs to create a joint color image. Resolution is preserved in this solution, although the three-chip color cameras are, as one would expect, significantly more expensive and therefore more rarely used in mobile robotics.

Both three-chip and single-chip color cameras suffer from the fact that photodiodes are much more sensitive to the near-infrared end of the spectrum. This means that the overall system detects blue light much more poorly than red and green. To compensate, the gain must be increased on the blue channel, and this introduces greater absolute noise on blue than on red and green. It is not uncommon to assume at least one to two bits of additional noise on the blue channel. Although there is no satisfactory solution to this problem today, over time the processes for blue detection have been improved, and we expect this positive trend to continue.

In color cameras, an additional control exists for *white balance*. Depending on the source of illumination in a scene (e.g., fluorescent lamps, incandescent lamps, sunlight, underwater filtered light, etc.), the relative measurements of red, green, and blue light that define pure white light will change dramatically. The human eye compensates for all such effects in ways that are not fully understood, but, the camera can demonstrate glaring inconsistencies in which the same table looks blue in one image, taken during the night, and yellow in another image, taken during the day. White balance controls enable the user to change the relative gains for red, green, and blue in order to maintain more consistent color definitions in varying contexts.

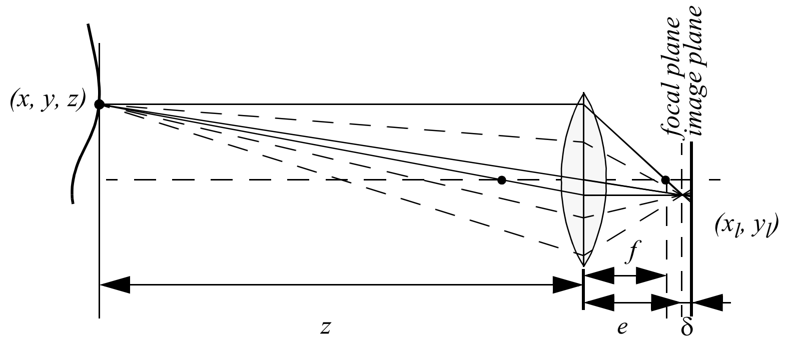


Figure 4.28
Depiction of the camera optics and its impact on the image. In order to get a sharp image, the image plane must coincide with the focal plane. Otherwise the image of the point (*x,y,z*) will be blurred in the image, as can be seen in the drawing above.

4.2.3 Image formation

Before we can intelligently analyze and manipulate images, we need to understand the image formation process that produced a particular image.

4.2.3.1 Optics

Once the light from the scene reaches the camera, it must still pass through the lens before reaching the sensor. Figure 4.28 shows a diagram of the most basic lens model, which is the thin lens. This lens is composed of a single piece of glass with very low, equal curvature on both sides. According to the lens law (which can be derived using simple geometric arguments on light ray refraction), the relationship between the distance to an object z and the distance behind the lens at which a focused image is formed e can be expressed as

$$\frac{1}{f} = \frac{1}{z} + \frac{1}{e}, \tag{4.33}$$

where f is the focal length. As you can perceive, this formula can also be used to estimate the distance to an object by knowing the focal length and the current distance of the image plane to the lens. This technique is called *depth from focus*.

If the image plane is located at distance e from the lens, then for the specific object voxel depicted, all light will be focused at a single point on the image plane and the object voxel will be *focused*. However, when the image plane is not at e, as is depicted in figure 4.28, then the light from the object voxel will be cast on the image plane as a *blur circle* (or *circle of confusion*). To a first approximation, the light is homogeneously distributed throughout this blur circle, and the radius R of the circle can be characterized according to the equation

$$R = \frac{L\delta}{2e}. \tag{4.34}$$

L is the diameter of the lens or aperture, and δ is the displacement of the image plane from the focal point.

Given these formulas, several basic optical effects are clear. For example, if the aperture or lens is reduced to a point, as in a pinhole camera, then the radius of the blur circle approaches zero. This is consistent with the fact that decreasing the iris aperture opening causes the *depth of field* to increase until all objects are in focus. Of course, the disadvantage of doing so is that we are allowing less light to form the image on the image plane, and so this is practical only in bright circumstances.

The second property that can be deduced from these optics equations relates to the sensitivity of blurring as a function of the distance from the lens to the object. Suppose the image plane is at a fixed distance 1.2 from a lens with diameter $L = 0.2$ and focal length $f = 0.5$. We can see from equation (4.34) that the size of the blur circle R changes proportionally with the image plane displacement δ. If the object is at distance $z = 1$, then from equation (4.33) we can compute $e = 1$, and therefore δ = 0.2. Increase the object distance to $z = 2$ and as a result δ = 0.533. Using equation (4.34) in each case, we can compute $R = 0.02$ and $R = 0.08$ respectively. This demonstrates high sensitivity for defocusing when the object is close to the lens.

In contrast, suppose the object is at $z = 10$. In this case we compute $e = 0.526$. But if the object is again moved one unit, to $z = 11$, then we compute $e = 0.524$. The resulting blur circles are $R = 0.117$ and $R = 0.129$, far less than the quadrupling in R when the obstacle is one-tenth the distance from the lens. This analysis demonstrates the fundamental limitation of depth from focus techniques: they lose sensitivity as objects move farther away (given a fixed focal length). Interestingly, this limitation will turn out to apply to virtually all visual ranging techniques, including depth from stereo (section 4.2.5) and depth from motion (section 4.2.6).

Nevertheless, camera optics can be customized for the depth range of the intended application. For example, a zoom lens with a very large focal length f will enable range resolution at significant distances, of course at the expense of field of view. Similarly, a large lens diameter, coupled with a very fast shutter speed, will lead to larger, more detectable blur circles.

Given the physical effects summarized by the above equations, one can imagine a visual ranging sensor that makes use of multiple images in which camera optics are varied (e.g., image plane displacement δ) and the same scene is captured (see figure 4.29). In fact, this

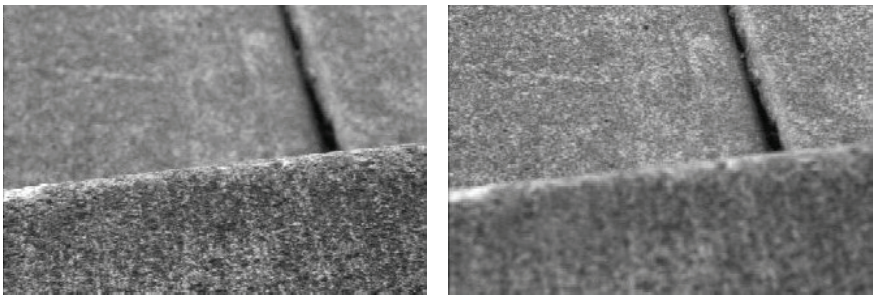


Figure 4.29
Two images of the same scene taken with a camera at two different focusing positions. Note the significant change in texture sharpness between the near surface and far surface. The scene is an outdoor concrete step.

approach is not a new invention. The human visual system uses an abundance of cues and techniques, and one system demonstrated in humans is depth from focus. Humans vary the focal length of their lens continuously at a rate of about 2 Hz. Such approaches, in which the lens optics are actively searched in order to maximize focus, are technically called *depth from focus* [241]. In contrast, *depth from defocus* means that depth is recovered using a series of images that have been taken with different camera geometries, and hence different focusing positions.

The depth from focus method is one of the simplest visual ranging techniques. To determine the range to an object, the sensor simply moves the image plane (via focusing) until maximizing the sharpness of the object. When the sharpness is maximized, the corresponding position of the image plane directly reports range. Some autofocus cameras and virtually all autofocus video cameras use this technique. Of course, a method is required for measuring the sharpness of an image or an object within the image.

An example application of depth-from-focus to robotics has been shown in [250], where the authors demonstrated obstacle avoidance in a variety of environments, as well as avoidance of concave obstacles such as steps and ledges.

4.2.3.2 Pinhole camera model

The pinhole camera, or *camera obscura*, has been the first example of camera in the history, which led to the invention of photography [27]. A pinhole camera has no lens, but a single very small aperture. In short, it is a lightproof box with a small hole in one side. Light from the scene passes through this single point and projects an inverted image on the opposite side of the box (figure 4.30). The working principle of this camera was already known as far back as the 4th century BC by the Greek Aristotle and Euclid and the Chinese Mozi.

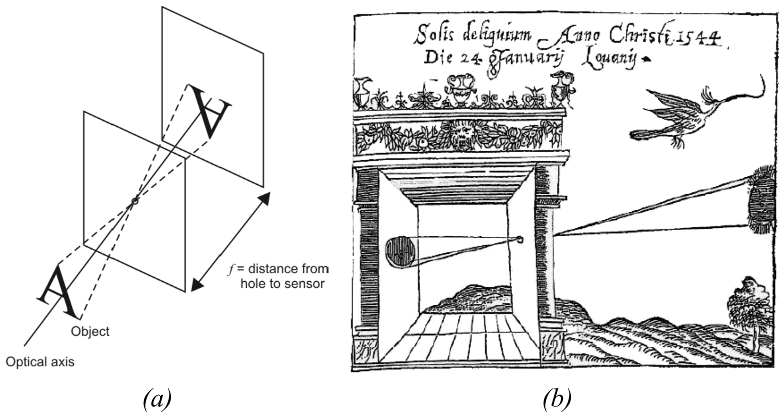

Solis deliquium Anno Christi 1544.
Die 24 Januarij Louanij
f = distance from hole to sensor
Object
Optical axis
(a)
(b)

Figure 4.30 (a) When $d \gg f$ and $d \gg L$ the camera can be modeled as a pinhole camera. (b) The *camera obscura* in a drawing from mathematician Reinerus Gemma-Frisius (1508–1555), who used this illustration in his book *De Radio Astronomica et Geometrica* (1545) to describe an eclipse of the sun at Louvain on January 24, 1544. It is thought to be the first published illustration of a camera obscura [27].

The pinhole projection model was also used as drawing aid by artists such as Leonardo da Vinci (1452–1519).

The importance of the pinhole camera is that its principle has also been adopted as a standard model for perspective cameras. This model can be derived directly from equation (4.33). In fact, notice that if we let $z \to \infty$, i.e. we adjust the lens (move the image plane) so that objects at infinity are in focus (i.e. $z \gg f$ and $z \gg L$), we get $e = f$, which is why we can think of a lens of focal length f as being equivalent (to a first approximation) to a pinhole a distance f from the focal plane (figure 4.31a).

When using the pinhole camera model, it is very important to remember that the pinhole corresponds to the center of the lens. This point is also commonly called *center of projection* of *optical center* (indicated with C in figure 4.31). The axis perpendicular to the *image plane* Π, which passes through the center of projection is called *optical axis*.

For convenience, the pinhole camera is commonly represented with the image plane between the center of projection and the scene (figure 4.31(b)). This is done for the image to preserve the same orientation as the object, that is, the image is not flipped. The intersection O between the optical axis and the image plane is called *principal point*.

As shown in figure 4.31b, observe that a camera does not measure distances but angles and therefore it can be thought as a *bearing* sensor.

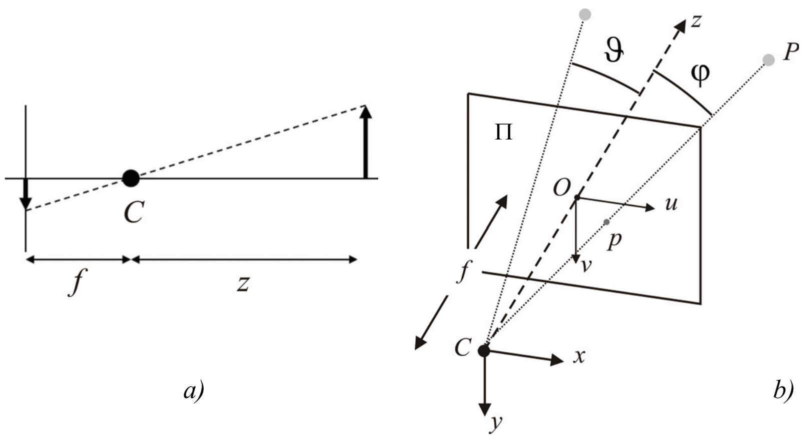


Figure 4.31 (a) Pinhole camera model used for representing standard perspective cameras. (b) The pinhole model is more commonly described with the image plane between the center of projection and the scene for the image to preserve the same orientation as the object.

4.2.3.3 Perspective projection

To describe analytically the *perspective projection* operated by the camera, we have to introduce some opportune reference system wherein we can express the 3D coordinates of the *scene point* P and the coordinates of its projection p on the image plane. We will first consider a simplified model and finally the general model.

Simplified model. Let (x, y, z) be the *camera reference frame* with origin in C and z-axis coincident with the optical axis. Assume also that the camera reference frame coincides with the *world* reference frame. This implies that the coordinates of the scene point P are already expressed in the camera frame.

Let us also introduce a two-dimensional reference frame (u, v) for the image plane Π with origin in O and the u and v axes aligned as x and y respectively as shown in figure 4.31b.

Finally, let $P = (x, y, z)$ and $p = (u, v)$. By means of simple considerations on the similarity of triangles, we can write

$$\frac{f}{z} = \frac{u}{x} = \frac{v}{y}, \tag{4.35}$$

and therefore

$$u = \frac{f}{z} \cdot x \ , \tag{4.36}$$

$$v = \frac{f}{z} \cdot y \ . \tag{4.37}$$

This is the *perspective projection*. The mapping from 3D coordinates to 2D coordinates is clearly nonlinear. However, using *homogeneous coordinates* instead allows us to obtain linear equations. Let

$$\tilde{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \text{ and } \tilde{P} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} , \tag{4.38}$$

be the homogeneous coordinates of p and P respectively. We will henceforth use the superscript ˜ to denote homogeneous coordinates[3]. The projection equation, in this simplified case, can be written as:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} . \tag{4.39}$$

Note that λ is equal to the third coordinate of P, which—in this special reference frame—coincides with the distance of the point to the plane xy. Note that this equation also shows that every image point is the projection of all infinite 3D points lying on the ray passing through the same image point and the center of projection (figure 4.31b). Therefore, using a single pinhole camera it is not possible to estimate the distance to a point, but we need two cameras (i.e., *stereo camera*, section 4.2.5.2).

General model. A realistic camera model that describes the transformation from 3D coordinates to pixel coordinates must also take into account

3. In homogeneous coordinates we denote 2D points in the image plane as (x_1, x_2, x_3) with $(x_1/x_3, x_2/x_3)$ being the corresponding Cartesian coordinates. Therefore, there is a one to many correspondence between Cartesian and homogeneous coordinates. Homogeneous coordinates can represent the usual Euclidean points plus the points at infinity, which are points with the last component equal to zero that do not have a Cartesian counterpart.

- the *pixelization*, that is, shape (size) of the CCD and its position with respect to the optical center,
- the rigid body transformation between the camera and the scene (i.e., world).

The pixelization takes into account the fact that:

1. The camera optical center has pixel coordinates (u_0, v_0) with respect to the upper left corner of the image, which is commonly assumed as origin of the image coordinate system. Note, the optical center in general does not correspond to the center of the CCD.
2. The coordinates of a point on the image plane are measured in pixels. Therefore, we must introduce a scale factor.
3. The shape of the pixel is in general assumed not perfectly squared and therefore we must use two different scale factors k_u and k_v along the horizontal and vertical directions respectively.
4. The u and v axes might not be orthogonal but misaligned of an angle θ. This models, for instance, the fact that the lens may not be parallel to the CCD.

The first three points are addressed by means of the translation of the optical center and the individual rescaling of the u and v axes:

$$u = k_u \frac{f}{z} \cdot x + u_0 \tag{4.40}$$

$$v = k_v \frac{f}{z} \cdot y + v_0 \ , \tag{4.41}$$

where (u_0, v_0) are the coordinates of the principal point, k_u (k_v) is the inverse of the effective pixel size along the u (v) direction and is measured in $pixel \cdot m^{-1}$.

After this update the perspective projection equations become:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} fk_u & 0 & u_0 & 0 \\ 0 & fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \ . \tag{4.42}$$

Observe that we can pose $\alpha_u = fk_u$ and $\alpha_v = fk_v$ which describe the focal lengths expressed in horizontal and vertical pixels respectively.

To take into account the fact that in general the world reference system $(x_w, y_w z_w)$ does not coincide with the camera reference system (x, y, z), we have to introduce the rigid body

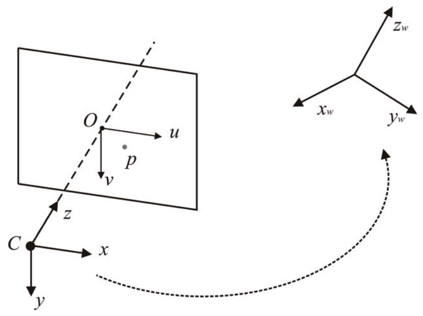


Figure 4.32 Coordinate change between camera and world reference frame.

transformation between the two reference frames (figure 4.32). Let us therefore introduce a coordinate change composed of a rotation R followed by a translation t, therefore

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + t . \tag{4.43}$$

Using this transformation, equation (4.42) can be rewritten as

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}, \tag{4.44}$$

or, using the homogeneous coordinates (4.38),

$$\lambda\tilde{p} = A[R|t]\tilde{P}_w , \tag{4.45}$$

where

$$A = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.46}$$

is the *intrinsic parameter matrix*.

As anticipated, the most general model also takes into consideration the possibility that the u and v axes are not orthogonal but are inclined of an angle θ. Therefore, the most general form for A is

$$A = \begin{bmatrix} \alpha_u & \alpha_u \cot\theta & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.47}$$

where $\alpha_u \cot\theta$ can be absorbed into a single parameter α_c.

α_c, α_u, α_{uv}, u_0, v_0 are called *camera intrinsic parameters.* The rotation and translation parameters R and t are called *camera extrinsic parameters*. The intrinsic and extrinsic parameters can be estimated using a procedure called *camera calibration* that we will shortly describe in section 4.2.3.4.

Radial distortion. The aforementioned image projection model assumes that the camera obeys a linear projection model where straight lines in the world result in straight lines in the image. Unfortunately, many wide-angle lenses have noticeable radial distortion, which manifests itself as a visible curvature in the projection of straight lines. An accurate model of the camera should therefore also take into account the radial distortion of the lens, especially for lenses with short focal length (i.e., large field of view) (figure 4.33).

The standard model of radial distortion is a transformation from the ideal coordinates (i.e., undistorted) (u, v) to the real observable coordinates (distorted) (u_d, v_d). Depending on the type of radial distortion, the coordinates in the observed images are displaced away (*barrel* distortion) or toward (*pincushion* distortion) the image center. The amount of distortion of the coordinates of the observed image is a nonlinear function of their radial distance r. For most lenses, a simple quadratic model of distortion produces good results:

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = (1 + k_1 r^2) \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}, \tag{4.48}$$

where

$$r^2 = (u - u_0)^2 + (v - v_0)^2 . \tag{4.49}$$

and k_1 is the radial distortion parameter, which can be estimated by camera calibration. The radial distortion parameter is also an intrinsic parameter of the camera.

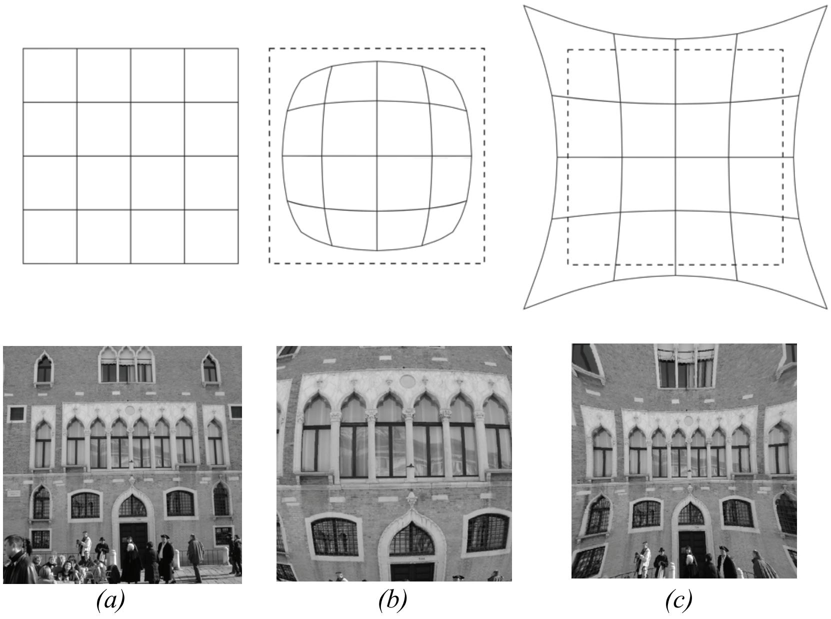


Figure 4.33 Example of radial lens distortion: (a) no distortion, (b) barrel distortion, (c) pincushion.

Sometimes the above simplified model does not model the true distortions produced by complex lenses accurately enough (especially at very wide angles). A more complete analytic model also includes *tangential distortions* and *decentering distortions* [48], but these will not be covered in this book. Fisheye lenses require a different model than traditional polynomial models of radial distortion and will be introduced in section 4.2.4.2.

4.2.3.4 Camera calibration

Calibration consists in measuring accurately the intrinsic and extrinsic parameters of the camera model. As these parameters govern the way the scene points are mapped to their corresponding image points, the idea is that by knowing the pixel coordinates of the image points $\tilde{p}$ and the 3D coordinates of the corresponding scene points $\tilde{P}$, it is possible to compute the unknown parameters A, R, and t by solving the perspective projection equation (4.45).

One of the first and most used camera calibration techniques was proposed in 1987 by Tsai [319]. Its implementation needs corresponding 3D point coordinates and 2D pixels coordinates in the image. It uses a two-stage technique to compute, first, the position and orientation and, second, the internal parameters of the camera.

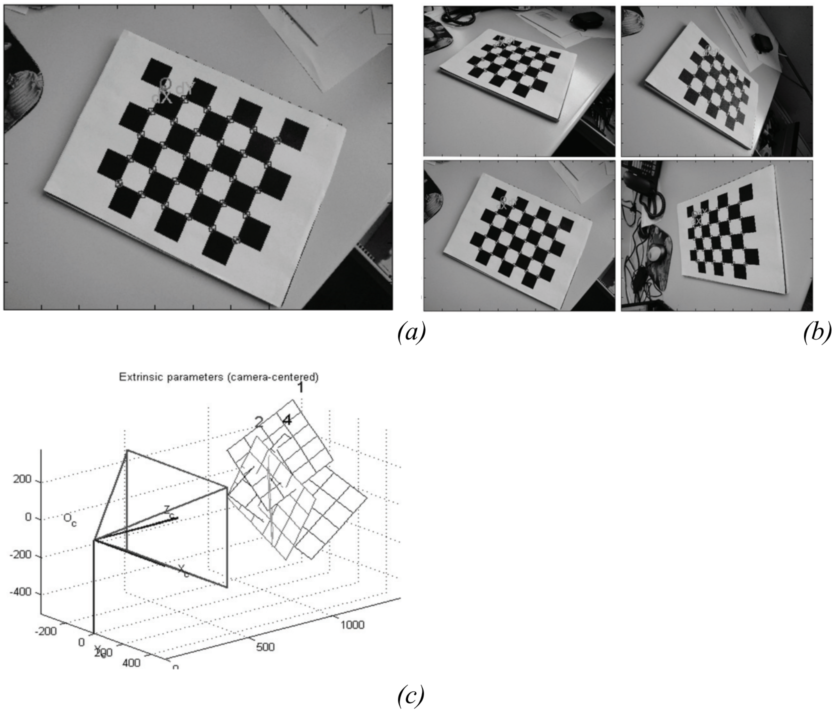


Figure 4.34 Pictures of the Camera Calibration Toolbox for Matlab developed by J. Y. Bouguet. (a) An example checkerboard-like pattern used in camera calibration with extracted corners. (b) Several pictures of the pattern with different positions and orientations. (c) The reconstructed position and orientations of the pattern after calibration.

However, in the last decade an alternative camera calibration technique has been proposed by Zhang [337] that, instead of a three-dimensional calibration object, uses a planar grid. The most common planar grid is a chessboard-like pattern due to the ease of extracting its corners, which are then used for calibration (figure 4.34). This method is known as *calibration from planar grid* and is very simple and practical to execute for both expert and non-expert users. The method requires the user to take several pictures of the pattern shown at different positions and orientations[4]. By knowing the 2D position of the corners on the real pattern and the pixel coordinates of their corresponding corners on each image, the intrinsic and extrinsic parameters (including radial and tangential distortion) are determined simultaneously by solving a least-square linear minimization followed by a nonlin-

4. Note that in this case the number of extrinsic parameters is different for each position of the grid while the intrinsic parameter are obviously the same.

ear refinement (i.e., Gauss-Newton). As pointed out by Zhang, the accuracy of the calibration results increases with the number of images used. It is also important that the images cover as much of the field of view of the camera as possible and that the range of orientations is wide.

This calibration method has been implemented in a very successful open source toolbox for Matlab (which can be downloaded for free [347]) that is also available in C in the open source Computer Vision Library (OpenCV) [343]. This toolbox has been used by thousands of users all around the world and is considered one of the most practical and easy-to-use camera calibration softwares for standard perspective cameras. In this section, we used the same model as the Matlab toolbox. This should facilitate the understanding and implementation of the interested reader. Alternatively, a complete list of all available camera calibration softwares can be found in [348].

4.2.4 Omnidirectional cameras

4.2.4.1 Introduction

In the previous section, we described the image formation of the pinhole camera, which is modeled as a perspective projection. However, there are projection systems whose geometry cannot be described using the conventional pinhole model because of the very high distortion introduced by the imaging device. Some of these systems are omnidirectional cameras.

An omnidirectional camera is a camera that provides wide field of view, at least more than 180 degrees. There are several ways to build an omnidirectional camera. Dioptric cameras use a combination of shaped lenses (e.g., fisheye lenses; see figure 4.35a) and typically can reach a field of view slightly larger than 180 degrees. Catadioptric cameras combine a standard camera with a shaped mirror, like a parabolic, hyperbolic, or elliptical mirror and are able to provide much more than 180 degrees field-of-view in elevation and 360 in the azimuthal direction. In figure 4.35b you can see an example catadioptric camera using a hyperbolic mirror. Finally, polydioptric cameras use multiple cameras with overlapping field of view (figure 4.35c) and so far are the only cameras able to provide a real omnidirectional (spherical) view (i.e., 4π radians).

Catadioptric cameras were first introduced in robotics in 1990 by Yagi and Kawato [333], who utilized them for localizing robots. Fisheye cameras started to spread over only in 2000 thanks to new manufacturing techniques and precision tools that led to an increase of their field of view up to 180 degrees. However, it is only since 2005 that these cameras have been miniaturized to the size of 1–2 centimeters, and their field of view has been increased up to 190 degrees (see, for instance, figure 4.36a).

Thanks to the camera miniaturization, to the recent developments in optics manufacturing, and to the decreasing prices in the cameras' market, catadioptric and dioptric omni-

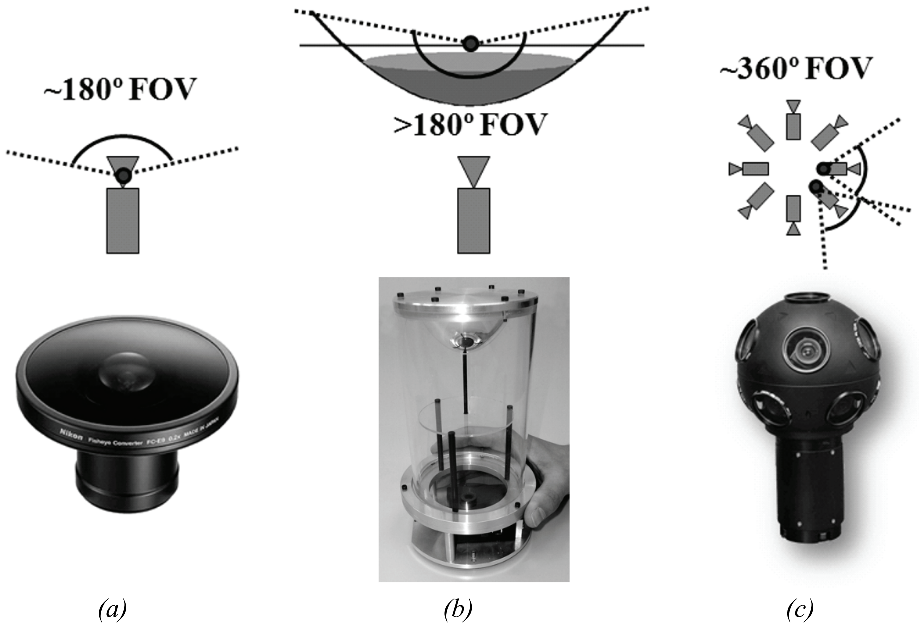


Figure 4.35 (a) Dioptric camera (e.g. fisheye); (b) catadioptric camera; (c) an example polydioptric camera produced by Immersive Media.

directional cameras are being more and more used in different research fields. Miniature dioptric and catadioptric cameras are now used by the automobile industry in addition to sonars for improving safety, by providing to the driver an omnidirectional view of the surrounding environment. Miniature fisheye cameras are used in endoscopes for surgical operations or on board microaerial vehicles for pipeline inspection as well as rescue operations. Other examples involve meteorology for sky observation.

Roboticists have also been using omnidirectional vision with very successful results on robot localization, mapping, and aerial and ground robot navigation [76, 80, 107, 278, 279, 307]. Omnidirectional vision allows the robot to recognize places more easily than with standard perspective cameras [276]. Furthermore, landmarks can be tracked in all directions and over longer periods of time, making it possible to estimate motion and build maps of the environment with better accuracy than with standard cameras, see figure 4.36 for some of examples of miniature omnidirectional cameras used on state-of-the-art micro aerial vehicles. Several companies, like Google, are using omnidirectional cameras to build photorealistic street views and three-dimensional reconstructions of cities along with texture. Two example omnidirectional images are shown in figure 4.37.

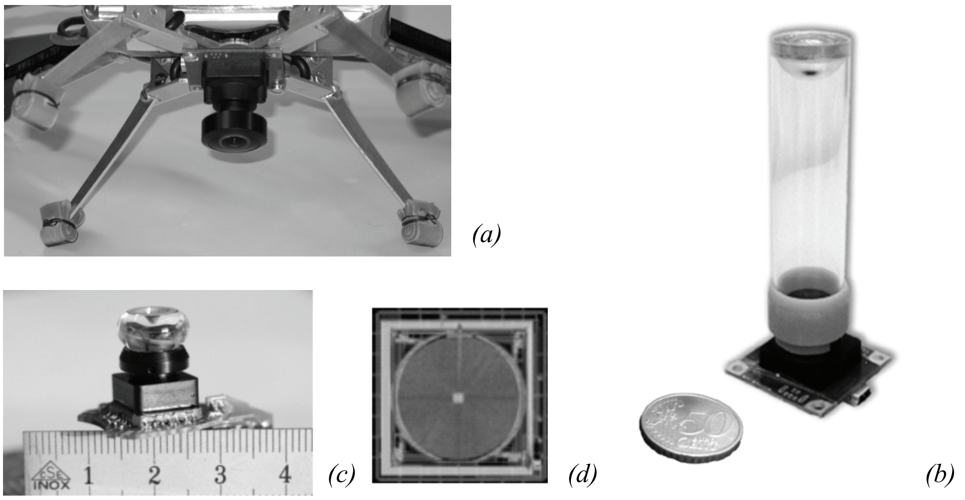


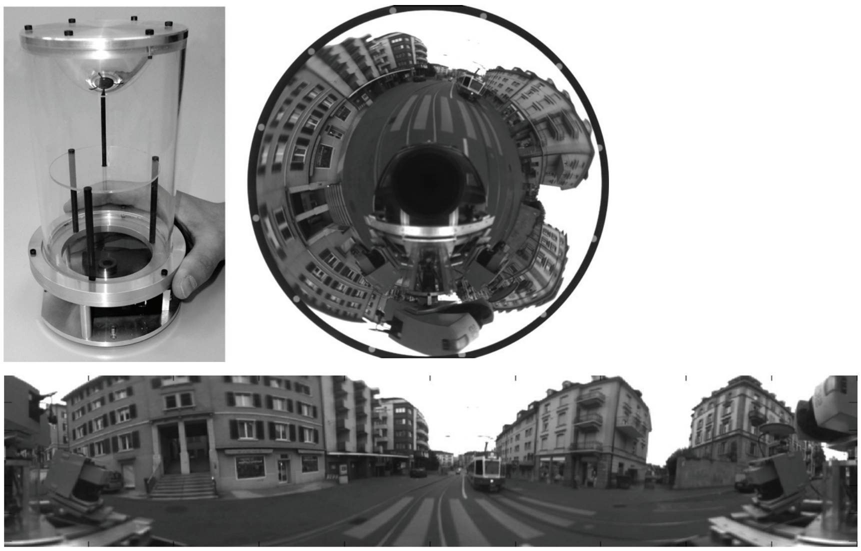
Figure 4.36 (a) The fisheye lens from Omnitech Robotics (www.omnitech.com) provides a field of view of 190 deg. This lens has a diameter of 1.7 cm. This camera has been used on the sFly autonomous helicopter at the ETH Zurich, (section 2.4.3) [76]. (b) A miniature catadioptric camera built at the ETH Zurich, which is also used for autonomous flight. It uses a spherical mirror and a transparent plastic support. The camera measures 2 cm in diameter and 8 cm in height. (c) The muFly camera built by CSEM, which is used on the muFly helicopter at the ETH Zurich (section 2.4.3). This is one of the smallest catadioptric cameras ever built. Additionally, it uses a polar CCD (d) where pixels are arranged radially.

In the next sections we will give an overview of omnidirectional camera models and calibration. For an in-depth study on omnidirectional vision, we refer the reader to [4, 15, 273].

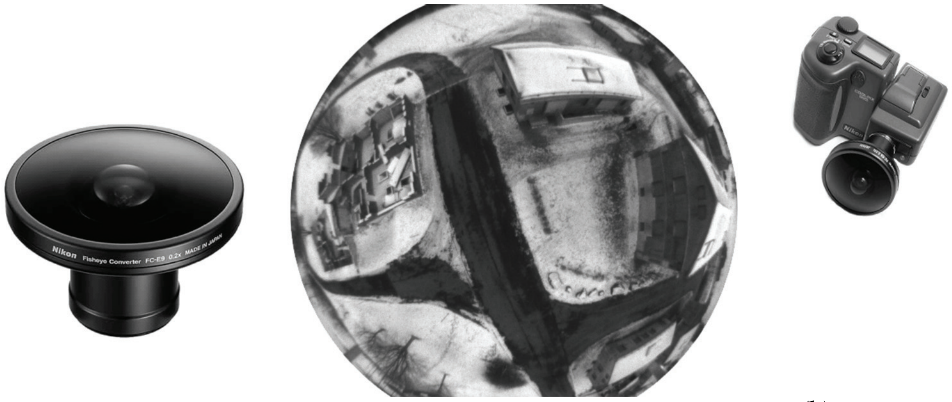
4.2.4.2 Central omnidirectional cameras

A vision system is said to be central when the optical rays to the viewed objects intersect in a single point in 3D called projection center or *single effective viewpoint* (figure 4.38). This property is called *single effective viewpoint property*. The perspective camera is an example of a central projection system because all optical rays intersect in one point, that is, the camera optical center.

All modern fisheye cameras are central, and hence, they satisfy the single effective viewpoint property. Central catadioptric cameras conversely can be built only by opportunely choosing the mirror shape and the distance between the camera and the mirror. As proven by Baker and Nayar [64], the family of mirrors that satisfy the single viewpoint property is the class of rotated (swept) conic sections, that is, hyperbolic, parabolic, and elliptical mirrors. In the case of hyperbolic and elliptical mirrors, the single view point



(a)



(b)

Figure 4.37 (a) A catadioptric omnidirectional camera using a hyperbolic mirror. The image is typically unwrapped into a cylindrical panorama. The field of view is typically 100 degrees in elevation and 360 degrees in azimuth. (b) Nikon fisheye lens FC-E8. This lens provides a hemispherical (180 deg) field of view.

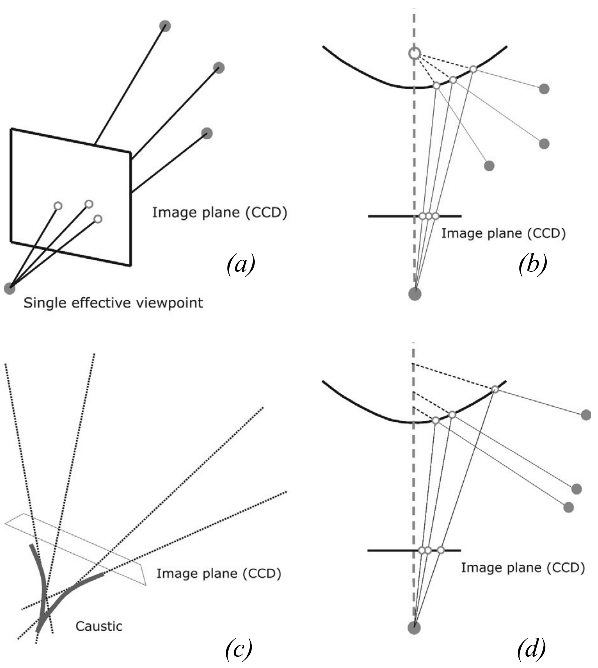


Figure 4.38 (a–b) Example of central cameras: perspective projection and catadioptric projection through a hyperbolic mirror. (c–d) Example of noncentral cameras: the envelope of the optical rays forms a *caustic*.

property is achieved by ensuring that the camera center (i.e., the pinhole or the center of the lens) coincides with one of the foci of the hyperbola (ellipse) (figure 4.39). In the case of parabolic mirrors, an orthographic lens must be interposed between the camera and the mirror, this makes it possible that parallel rays reflected by the parabolic mirror converge to the camera center (figure 4.39).

The reason a single effective viewpoint is so desirable is that it allows us to generate geometrically correct perspective images from the pictures captured by the omnidirectional camera (figure 4.40). This is possible because, under the single view point constraint, every pixel in the sensed image measures the irradiance of the light passing through the viewpoint in one particular direction. When the geometry of the omnidirectional camera is known, that is, when the camera is calibrated, one can precompute this direction for each pixel. Therefore, the irradiance value measured by each pixel can be mapped onto a plane at any distance from the viewpoint to form a planar perspective image. Additionally, the image can be mapped on to a sphere centered on the single viewpoint, that is, spherical projection (figure 4.40, bottom).

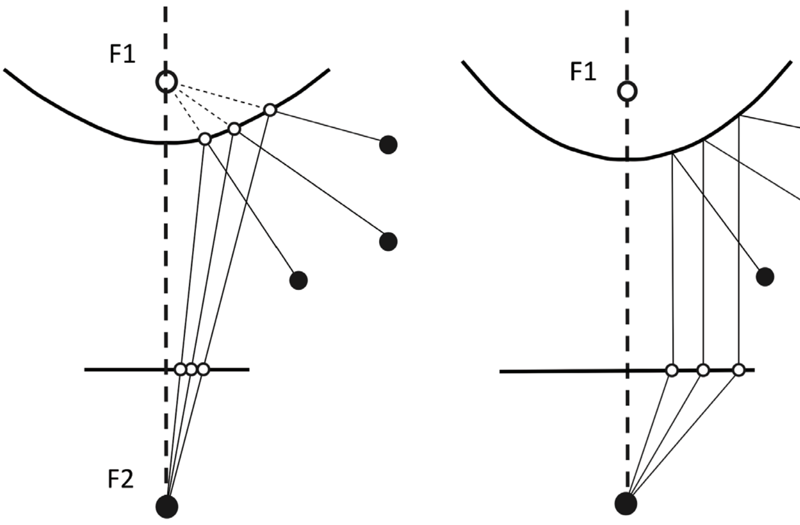


Figure 4.39 Central catadioptric cameras can be built by using hyperbolic and parabolic mirrors. The parabolic mirror requires the use of an orthographic lens.

Another reason why the single view point property is so important is that it allows us to apply the well known theory of *epipolar geometry* (see section 4.2.6.1), which easily allows us to perform *structure from stereo* (section 4.2.5) and *structure from motion* (section 4.2.6). As we will see, epipolar geometry holds for any central camera, both perspective and omnidirectional. Therefore, in those sections we will not make any distinction about the camera.

4.2.4.3 Omnidirectional camera model and calibration

Intuitively, the model of an omnidirectional camera is a little more complicated than a standard perspective camera. The model should indeed take into account the reflection operated by the mirror in the case of a catadioptric camera or the refraction caused by the lens in the case of a fisheye camera. Because the literature in this field is quite large, here we review two different projection models that have become standards in omnidirectional vision and robotics. Additionally, Matlab toolboxes have been developed for these two models, which are used worldwide by both specialists and non-experts.

The first model is known as the *unified projection model for central catadioptric cameras*. It was developed in 2000 by Geyer and Daniilidis [137] (later refined by Barreto and Araujo [66]), who have the merit of having proposed a model that encompasses all three types of central catadioptric cameras, that is, cameras using hyperbolic, parabolic, or elliptical mirror. This model was developed specifically for central catadioptric cameras and is

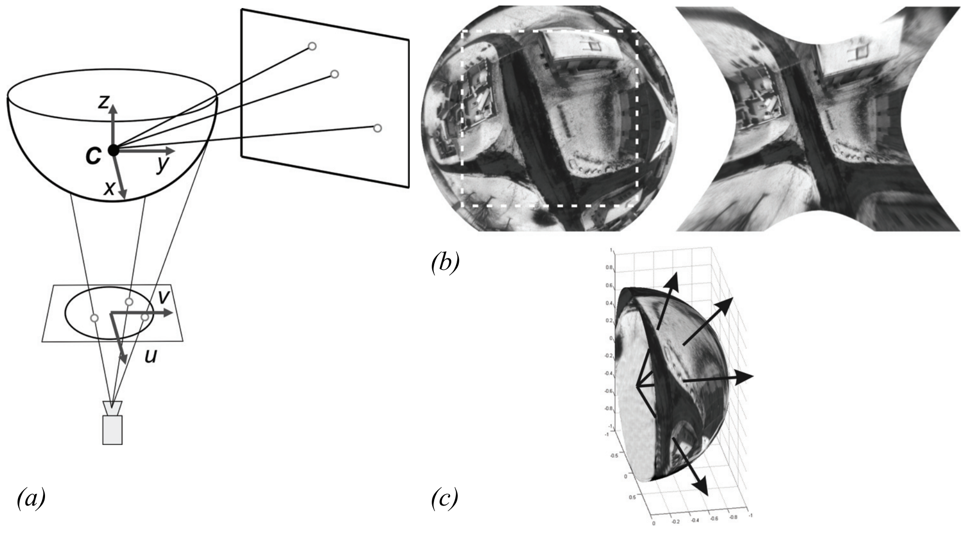


Figure 4.40 Central cameras allows us to remap regions of the omnidirectional image into a perspective image. This can be done straightforwardly by intersecting the optical rays with a plane specified arbitrarily by the user (a). Of course we cannot project the whole image onto a plane but only subregions of it (b). Another possible projection is that onto a sphere (c).

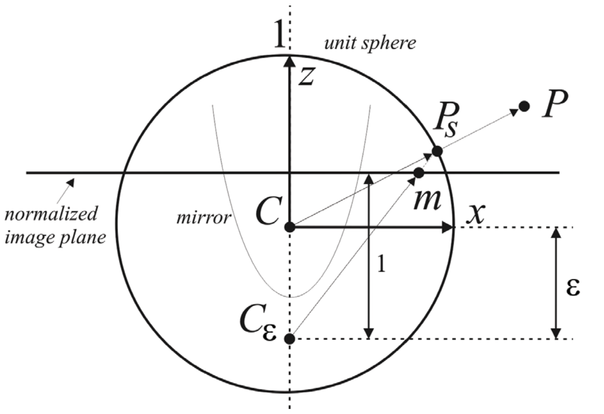
not valid for fisheye cameras. The approximation of a fisheye lens model by a catadioptric one is usually possible, however, with limited accuracy only [335].

Conversely, the second model unifies both central catadioptric cameras and fisheye cameras under a general model also known as Taylor model. It was developed in 2006 by Scaramuzza et al. [274, 275] and has the advantage that both catadioptric and dioptric cameras can be described through the same model, namely a Taylor polynomial.

Unified model for central catadioptric cameras. With their landmark paper from 2000, Geyer and Daniilidis showed that every catadioptric (parabolic, hyperbolic, elliptical) and standard perspective projection is equivalent to a projective mapping from a sphere, centered in the single viewpoint, to a plane with the projection center on the perpendicular to the plane and distant ε from the center of the sphere. This is summarized in figure 4.41.

As we did for the perspective camera, the goal is again to find the relation between the viewing direction to the scene point and the pixel coordinates of the corresponding image point. The projection model of Geyer and Daniilidis follows a four-step process. Let again $P = (x, y, z)$ be a scene point in the mirror reference frame[5] centered in C (figure 4.41).

1. The first step consists in projecting the scene point onto the unit sphere; therefore:



Mirror type	ε
Parabola	1
Hyperbola	$\frac{d}{\sqrt{d^2+4l^2}}$
Ellipse	$\frac{d}{\sqrt{d^2+4l^2}}$
Perspective	0

Figure 4.41 Unified projection model for central catadioptric cameras of Geyer and Daniilidis.

$$P_s = \frac{P}{\|P\|} = (x_s, y_s, z_s) . \tag{4.50}$$

2. The point coordinates are then changed to a new reference frame centered in $C_\varepsilon = (0, 0, -\varepsilon)$; therefore:

$$P_\varepsilon = (x_s, y_s, z_s + \varepsilon) . \tag{4.51}$$

Observe that ε ranges between 0 (planar mirror) and 1 (parabolic mirror). The correct value of ε can be obtained knowing the distance *d* between the foci of the conic and the latus rectum[6] *l* as summarized in the table of figure 4.41.

3. P_ε is then projected onto the normalized image plane distant 1 from C_ε; therefore,

$$\tilde{m} = (x_m, y_m, 1) = \left(\frac{x_s}{z_s+\varepsilon}, \frac{y_s}{z_s+\varepsilon}, 1\right) = g^{-1}(P_s) . \tag{4.52}$$

4. Finally, the point $\tilde{m}$ is mapped to the camera image point $\tilde{p} = (u, v, 1)$ through the intrinsic parameter matrix A; therefore,

5. For convenience we assume that the mirror axis of symmetry is perfectly aligned with the camera optical axis. We also assume that the *x*-*y* axes of the camera and mirror are aligned. Therefore, the camera and mirror reference frames differ only by a translation along *z*.

6. The latus rectum of a conic section is the chord through a focus parallel to the conic section directrix.

$$\tilde{p} = A \cdot \tilde{m}, \tag{4.53}$$

where A is given by (4.47), that is,

$$A = \begin{bmatrix} \alpha_u & \alpha_u \cot\theta & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.54}$$

It is easy to show that function g^{-1} is bijective and that its inverse g is given by[7]:

$$P_s = g(m) \sim \begin{bmatrix} x_m \\ y_m \\ 1 - \varepsilon \dfrac{x_m^2 + y_m^2 + 1}{\varepsilon + \sqrt{1 + (1 - \varepsilon^2)(x_m^2 + y_m^2)}} \end{bmatrix}, \tag{4.55}$$

where ~ indicates that g is proportional to the quantity on the right-hand side. To obtain the scale factor, it is sufficient to normalize $g(m)$ onto the unit sphere.

Observe that equation (4.55) is the core of the projection model of central catadioptric cameras. It expresses the relation between the point m on the normalized image plane and the unit vector P_s in the mirror reference frame. Note that in the case of planar mirror, we have $\varepsilon = 0$ and (4.55) becomes the projection equation of perspective cameras $P_s \sim (x_m, y_m, 1)$.

This model has proved to be able to describe accurately all central catadioptric cameras (parabolic, hyperbolic, and elliptical mirror) and standard perspective cameras. An extension of this model for fisheye lenses was proposed in 2004 by Ying and Hu [335]. However, the approximation of a fisheye camera through a catadioptric one works only with limited accuracy. This is mainly due because, while the three types of central catadioptric cameras can be represented through an exact parametric function (parabola, hyperbola, ellipse), the projective models of fisheye vary from camera to camera and depend on the lens field-of-view. To overcome this problem, a new unified model has been proposed, which will be described in the next section.

7. Equation (4.55) can be obtained by inverting (4.52) and imposing the constraint that P_s must lie on the unit sphere and, thus, $x_s^2 + y_s^2 + z_s^2 = 1$. From this constraint you will then get an expression of z_s as a function of ε, x_m, and y_m. More details can be found in [66].

Unified model for catadioptric and fisheye cameras. This unified model was proposed by Scaramuzza et al. in 2006 [274, 275]. The main difference with the previous model lies in the choice of the function g. To overcome the lack of knowledge of a parametric model for fisheye cameras, the authors proposed the use of a Taylor polynomial, whose coefficients and degree are found through the calibration process. Accordingly, the relation between the normalized image point $\tilde{m} = (x_m, y_m, 1)$ and the unit vector P_s in the fisheye (mirror) reference frame can be written as:

$$P_s = g(m) \sim \begin{bmatrix} x_m \\ y_m \\ a_0 + a_2\rho^2 + \ldots + a_N\rho^N \end{bmatrix}, \tag{4.56}$$

where $\rho = \sqrt{x_m^2 + y_m^2}$. As you have probably noticed, the first-order term (i.e., $a_1\rho$) of the polynomial is missing. This follows from the observation that the first derivative of the polynomial calculated at $\rho = 0$ must be null for both catadioptric and fisheye cameras (this is straightforward to verify for catadioptric cameras by differentiating [4.55]). Also observe that because of its polynomial nature, this expression can encompass catadioptric, fisheye, and perspective cameras. This can be done by opportunely choosing the degree of the polynomial. As highlighted by the authors, polynomials of order three or four are able to model very accurately all catadioptric cameras and many types of fisheye cameras available on the market. The applicability of this model to a wide range of commercial cameras is at the origin of its success.

Omnidirectional camera calibration. The calibration of omnidirectional cameras is similar to that for calibrating standard perspective cameras, which we have seen in section 4.2.3.4. Again, the most popular methods take advantage of planar grids that are shown by the user at different positions and orientations. For omnidirectional cameras, it is very important that the calibration images are taken all around the camera and not on a single side only. This in order to compensate for possible misalignments between the camera and mirror.

It is worth to mention three open-source calibration toolboxes currently available for Matlab, which differ mainly for the projection model adopted and the type of calibration pattern.

- The toolbox of Mei uses checkerboard-like images and takes advantage of the projection model of Geyer and Daniilidis discussed earlier. It is particularly suitable for catadioptric cameras using hyperbolic, parabolic, folded mirrors, and spherical mirrors. Mei's toolbox can be downloaded from [349], while the theoretical details can be found in [212].

- The toolbox of Barreto uses line images instead of checkerboards. Like the previous toolbox, it also uses the projection model of Geyer and Daniilidis. It is particularly suitable for parabolic mirrors. The toolbox can be downloaded from [350], while the theoretical details can be found in [67] and [68].
- Finally, the toolbox of Scaramuzza uses checkerboard-like images. Contrary to the previous two, it takes advantage of the unified Taylor model for catadioptric and fisheye cameras developed by the same author. It works with catadioptric cameras using hyperbolic, parabolic, folded mirrors, spherical, and elliptical mirrors. Additionally, it works with a wide range of fisheye lenses available on the market—like Nikon, Sigma, Omnitech-Robotics, and many others—with field of view up to 195 degrees. The toolbox can be downloaded from [351], while the theoretical details can be found in [274] and [275]. Contrary to the other two, this toolbox features an automatic calibration process. In fact, both the center of distortion and the calibration points are detected automatically without any user intervention.

4.2.5 Structure from stereo

4.2.5.1 Introduction

Range sensing is extremely important in mobile robotics, since it is a basic input for successful obstacle avoidance. As we have seen earlier in this chapter, a number of sensors are popular in robotics explicitly for their ability to recover depth estimates: ultrasonic, laser rangefinder, time-of-flight cameras. It is natural to attempt to implement ranging functionality using vision chips as well.

However, a fundamental problem with visual images makes rangefinding relatively difficult. Any vision chip collapses the 3D world into a 2D image plane, thereby losing depth information. If one can make strong assumptions regarding the size of objects in the world, or their particular color and reflectance, then one can directly interpret the appearance of the 2D image to recover depth. But such assumptions are rarely possible in real-world mobile robot applications. Without such assumptions, a single picture does not provide enough information to recover spatial information.

The general solution is to recover depth by looking at *several* images of the scene to gain more information, hopefully enough to at least partially recover depth. The images used must be different, so that taken together they provide additional information. They could differ in camera geometry—such as the focus position or lens iris—yielding depth from focus (or defocus) techniques that we have described in section 4.2.3.1. An alternative is to create different images, not by changing the camera geometry, but by changing the camera viewpoint to a different camera position. This is the fundamental idea behind *structure from stereo* (i.e., *stereo vision*) and *structure from motion* that we will present in the next sections. As we will see, stereo vision processes two distinct images taken at the same time and assumes that the relative pose between the two cameras is known. Structure-from-motion conversely processes two images taken with the same or a different camera at different times and from different unknown positions; the problem consists in recovering both the relative motion between the views and the depth. The 3D scene that we want to reconstruct is usually called *structure*.

4.2.5.2 Stereo vision

Stereopsis (from *stereo* meaning solidity, and *opsis* meaning vision or sight) is the process in visual perception leading to the sensation of depth from the two slightly different projections of the world onto the retinas of the two eyes. The difference in the two retinal images is called horizontal *disparity*, retinal disparity, or binocular disparity. The differences arise from the eyes' different positions in the head. It is the disparity that makes our brain fuse (perceive as a single image) the two retinal images making us perceive the object as a one and solid. To have a clearer understanding of what disparity is, as a simple test, hold your finger vertically in front of you and close each eye alternately. You will see that the finger jumps from left to right. The distance between the left and right appearance of the finger is the disparity. The same phenomenon is visible in the image pair shown in figure 4.48, in which the foreground objects shift left and right relative to the background.

Computational stereopsis, or stereo vision, is the process of obtaining depth information from a pair of images coming from two cameras which look at the same scene from different positions. In stereo vision we can identify two major problems:

1. the correspondence problem
2. 3D reconstruction

The first consists in matching (pairing) points of the two images which are the projection of the same point in the scene. These matching points are called *corresponding points* or *correspondences* (figure 4.45a). This will be clarified later on. Determining the corresponding points is made possible based on the assumption that the two images differ only slightly and therefore a feature in the scene appears similar in both images. Based only of this assumption, however, there might be many possible false matches. As we will see, this problem can be overcome by introducing an additional constraint which makes the correspondence matching feasible. This constraint is called *epipolar constraint* (section 4.2.6.1) and states that the correspondent of a point in an image lies on a line (called *epipolar line*) in the other image (figure 4.45b). Because of this constraint, we will see that the correspondence search becomes one-dimensional instead of two-dimensional.

Knowing the correspondences between the two images, knowing the relative orientation and position of the two cameras, and knowing the intrinsic parameters of the two cameras, it is possible to reconstruct the scene points (i.e., the structure). This process of reconstruction requires the prior calibration of the stereo camera; that is, we need to calibrate the two cameras separately for estimating their extrinsic parameters, but we also need to determine their extrinsic parameters, i.e. the camera relative position.

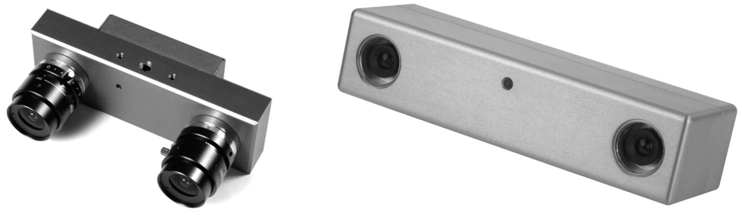


Figure 4.42 (Left) The STH-MDCS3 form Videre Design uses CMOS sensors, a baseline of 9 cm, an image resolution of 1280×960 at 7.5 frames per second (fps), or 640×480 at 30 fps. (Right) The Bumblebee2 from Point Grey uses CCD sensors, a baseline of 12 cm, an image resolution of 1024×768 at 20 frames per second (fps), or 640×480 at 48 fps.

The theory of stereo vision has been well understood for years, while the engineering challenge of creating a practical stereo-vision sensor has been formidable [21, 43, 44]. Example of commercially available stereo cameras are shown in figure 4.42.

Basic case. First, we consider a simplified case in which two cameras have the same orientation and are placed with their optical axes parallel, at a separation of b (called *baseline*), shown in figure 4.43.

In this figure, a point on the object is described as being at coordinate (x, y, z) with respect to the origin located in the left camera lens. The image coordinate in the left and right image are (u_l,v_l) and (u_r,v_r) respectively. From figure 4.43a and using equations (4.36) and (4.37), we can write

$$\frac{f}{z} = \frac{u_l}{x} , \tag{4.57}$$

$$\frac{f}{z} = \frac{-u_r}{b-x}, \tag{4.58}$$

from which we obtain

$$z = b\frac{f}{u_l - u_r} , \tag{4.59}$$

where the *difference* in the image coordinates, $u_l - u_r$ is called *disparity*. This is an important term in stereo vision, because it is only by measuring disparity that we can recover depth information. Observations from this equation are as follows:

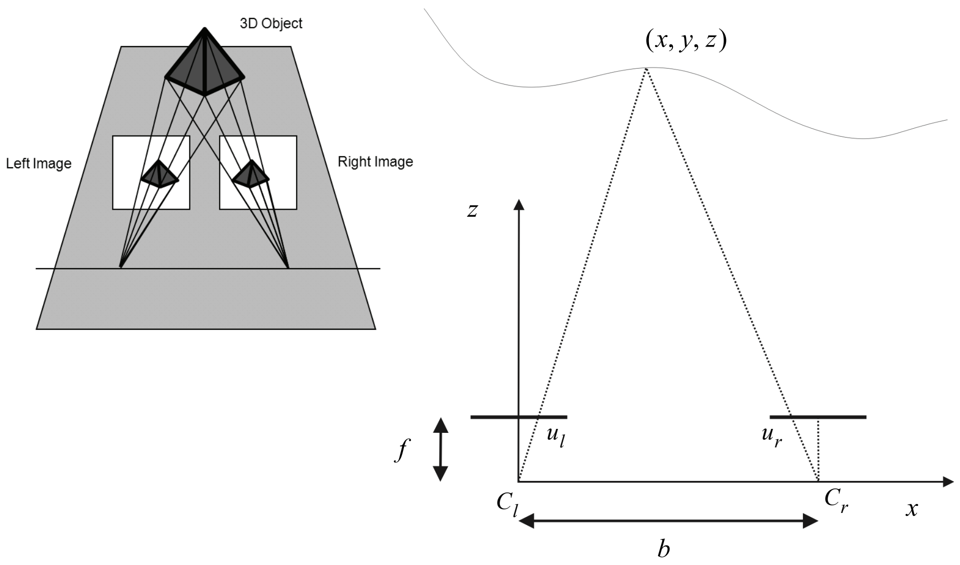


Figure 4.43
Idealized camera geometry for stereo vision. The cameras are assumed be identical (i.e., identical focal lengths and image resolution); furthermore, they are assumed to be perfectly aligned on the horizontal axis.

- Distance is inversely proportional to disparity. The distance to near objects can therefore be measured more accurately than that to distant objects, just as with depth from focus techniques. In general, this is acceptable for mobile robotics, because for navigation and obstacle avoidance closer objects are of greater importance.
- Disparity is proportional to b . For a given disparity error, the accuracy of the depth estimate increases with increasing baseline b .
- As b is increased, because the physical separation between the cameras is increased, some objects may appear in one camera but not in the other. This is due to the field of view of the cameras. Such objects by definition will not have a disparity and therefore will not be ranged.
- If the baseline b is unknown, it is possible to reconstruct the scene point only *up to a scale*. This is the case in structure-from-motion (section 4.2.6).
- A point in the scene visible to both cameras produces a pair of image points known as a *conjugate pair*, or *correspondence pair* (figure 4.44a). Given one member of the conjugate pair, we know that the other member of the pair lies somewhere along a line known

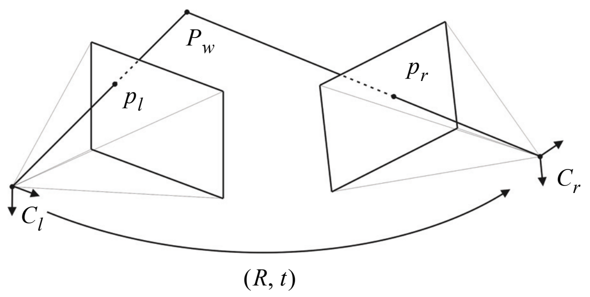


Figure 4.44 Stereo vision: general case.

as *epipolar line*. In the case depicted in figure 4.43a, because the cameras are perfectly aligned with one another, the epipolar lines are horizontal lines (i.e., along the x direction). The concept of epipolar line will be explained later on in this section.

General case. The assumption of perfectly aligned cameras is normally violated in practice. In fact, even the most expensive stereo cameras available in the market do not assume this model. Indeed, two exactly identical cameras do not exist. There will be always differences in the focal length due to manufacturing but, especially, even if such identical cameras could exist, we would never be sure that they are perfectly aligned. The situation is even more complicated by the fact that the internal orientation of the CCD in the camera package is unknown. Ideally it is aligned, but in practice the CCD cannot be considered perfectly aligned. Therefore, the general stereo vision model assumes that the two cameras are different and not aligned (figure 4.44) but requires that the relative position and orientation of the two cameras is known. If the relative position is not known, the stereo camera must be calibrated using the checkerboard-based calibration treated in section 4.2.3.4. Fortunately, the previously mentioned toolbox for calibrating the camera intrinsic parameters [347] allows the user to calibrate stereo cameras as well.

So, let us assume that the two cameras have been previously calibrated. Therefore, the intrinsic parameter matrices A_l and A_r (see equation 4.47) for the left and right camera are known, and the camera extrinsic parameters—i.e. the rotations R_l, R_r and translations t_l, t_r of the two cameras with the respect to the world coordinate system—are also known. In stereo vision, it is a common practice to assume the origin of the world coordinate system in the left camera. Thus, we can write $R_l = I$ and $R_r = R$. This allows us to write the equations of perspective projection for the two cameras as:

$$\lambda_l \tilde{p}_l = A_l[I|0]\tilde{P}_w \text{ (for the left camera),} \tag{4.60}$$

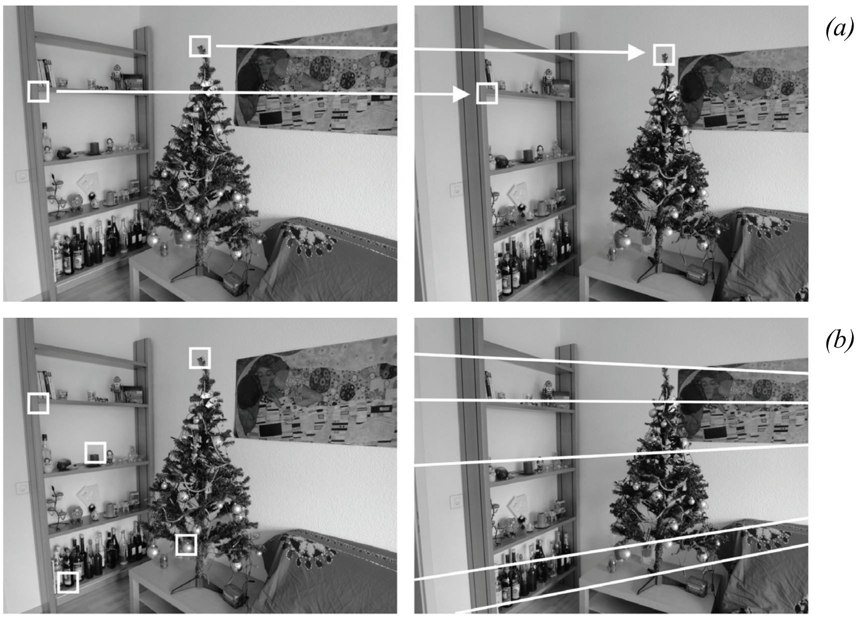


Figure 4.45 A stereo pair. Corresponding points are projections of the same scene point. Because of the epipolar constraint, conjugate points can be searched along the epipolar lines. This heavily reduces the computational cost of the correspondence search: from a two-dimensional search it becomes a one-dimensional search problem.

$$\lambda_r \tilde{p}_r = A_r[R|t]\tilde{P}_w \text{ (for the right camera),} \tag{4.61}$$

where $\tilde{p}_l = [u_l, v_l, 1]^T$ and $\tilde{p}_r = [u_r, v_r, 1]^T$ are the image points (in homogeneous coordinates) corresponding to the world point $\tilde{P}_w = [x, y, z, 1]^T$ (in homogeneous coordinates) in the left and right camera respectively. λ_l and λ_r are the depth factors. Observe that (4.60) and (4.61) actually contribute three equations each. Therefore, we have a system of six equations in five unknowns, three for the world point $P_w = (x, y, z)$ and two for depth factors, i.e. λ_l and λ_r. The system is overdetermined and can be solved either linearly, using least-squares, or nonlinearly by computing the 3D point that minimizes distances between the two light rays passing through $\tilde{p}_l$ and $\tilde{p}_r$. The solution of these two equations is left as an exercise to the reader in section 4.8.

Correspondence problem. Using the preceding equations requires us to have identified the conjugate pair p_l and p_r in the left and right camera images, which originates from the same scene point $\tilde{P}_w$ (figure 4.45a). This fundamental challenge is called the *correspon-*

dence problem. Intuitively, the problem is: given two images of the same scene from different perspectives, how can we identify the same object points in both images? For every such identified object point, we will then be able to recover its 3D position in the scene.

The correspondence search is based on the assumption that the two images of the same scene do not differ too much, that is, a feature in the scene is supposed to appear very similar in both images. Using an opportune image similarity metric (see section 4.3.3), a given point in the first image can be paired with one point in the second image. The problem of *false correspondences* makes the correspondence search challenging. False correspondences occur when a point is paired to another that is not its real conjugate. This is because the assumption of image similarity does not hold very well, for instance if the part of the scene to be paired appears under different illumination or geometric conditions. Other problems that make the correspondence search difficult are:

- *Occlusions*: the scene is seen by two cameras at different viewpoints and therefore there are parts of the scene that appear only in one of the images. This means, there exist points in one image which do not have a correspondent in the other image.
- *Photometric distortion*: there are surfaces in the scene which are nonperfectly *lambertian*, that is, surfaces whose behavior is partly specular. Therefore, the intensity observed by the two cameras is different for the same point in the scene as more as the cameras are farther apart.
- *Projective distortion*: because of the perspective distortion, an object in the scene is projected differently on the two images, as more as the cameras are farther apart.

Some constraints can, however, be exploited for improving the correspondence search, which are:

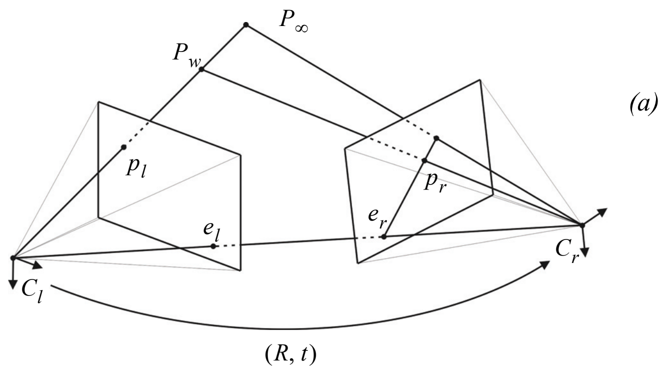
- *Similarity constraint*: a feature in the image appears similar in the other image.
- *Continuity constraint*: far from the image borders, the depth of the scene points along a continuous surface varies continuously. This constraint clearly limits the *gradient of disparity*.
- *Unicity*: a point of the first image can be paired only with a single point in the other image, and vice versa (it fails in presence of occlusions, specularities, and transparency).
- *Monotonic order*: if point p_l in the left image is the correspondent of p_r in the right image, then the correspondent of a point on the right (left) of p_l can only be found on the right (left) of p_r. This is valid only for points that lie on an opaque object.
- *Epipolar constraint*: the correspondent of a point in the left image can only be found along a line in the right image, which is called *epipolar line* (figure 4.45b). As a matter of fact, this is the most important constraint and will be explained later on.

The methods for searching correspondences can be distinguished into two categories:

- *Area-based*: these algorithms consider a small patch (window) in one image and look for the most similar patch in the second image by means of an appropriate *correlation* measure. This search is done for every pixel and allows us to obtain a *dense* reconstruction. However, in uniform regions—that is, poor texture—these methods fail. There exist different techniques to measure the similarity between image patches for stereo matching. The most used are the *Sum of Absolute Differences* (SAD), the *Sum of Squared Differences* (SSD), the *Normalized Cross Correlation* (NCC), and the *Census Transform*. An overview of some of these algorithms is given in section 4.3.3. Finally, observe that the search for correspondences is a two-dimensional search: the most similar of a patch in the left image must be searched across all rows and columns of the right image. As we will see in the next section, the search can be reduced to only one line, the epipolar line, thus reducing the dimensionality of the search from two to one (figure 4.45b).
- *Feature-based*: these algorithms extract salient features from the images, which are possibly stable with respect to change of view point. The matching process is applied to the attributes associated to the features. Edges, corners, line segments, and blobs are some of the features that can be used. They do not have to correspond necessarily to a well defined geometric entity. An exhaustive overview on feature extraction is given is section 4.5. Feature-based stereo matching algorithms are faster and more robust than area-based methods but provide only sparse depth maps, which then need to be interpolated.

Epipolar geometry. Given a pixel in one image (say the left image), how can we compute its correspondence with the correct pixel in the other image? As we anticipated in the previous section, one way would be to search the correspondences across all pixels of the second image. In the case of stereo matching, however, we have some information available, namely the relative position and the calibration parameters of the two cameras. This information allows us to reduce the search from two dimensions to an only one dimension. Figure 4.46a shows how a pixel point p_l in one image projects to an epipolar line segment in the other image. The segment is bounded at one end by the projection of P_∞ (the original viewing ray at infinity) and at the other end by the projection of C_l into the second camera, which is known as the epipole e_r. By projecting the epipolar line in the second image back into the first image, we get another line which is bounded by the other corresponding epipole e_l. Notice that two corresponding epipolar lines (figure 4.46b) originate from the intersection of the two image planes with the epipolar plane that passes through the camera centers C_l and C_r and the scene point P_w.

To compute the equation of the epipolar line, we must project the optical ray passing through p_l and C_l to the second image. This is straightforward. The equation of the optical



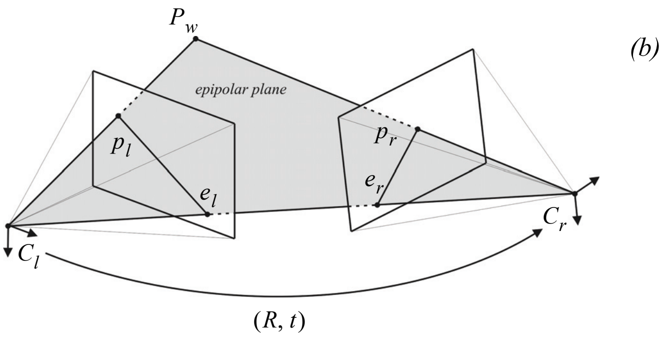


Figure 4.46 Epipolar geometry: (a) epipolar line segment corresponding to one ray; (b) corresponding set of epipolar lines and their epipolar plane.

ray passing through p_l and C_l can be obtained from the perspective projection equation (4.60), which we rewrite here as:

$$\lambda_l \tilde{p}_l = \lambda_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = A_l [I|0] \tilde{P}_w = A_l P_w = A_l \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{4.62}$$

and therefore the line passing through p_l and C_l has equation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \lambda_l A_l^{-1} \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix}, \tag{4.63}$$

which we can rewrite in a more compact form as:

$$P_w = \lambda_l A_l^{-1} \tilde{p}_l . \tag{4.64}$$

Finally, to find the equation of the epipolar line, we just project this line onto the second image using the perspective projection equation (4.61):

$$\lambda_r \tilde{p}_r = A_r [R|t] \tilde{P}_w = A_r R P_w + A_r t , \tag{4.65}$$

and therefore, using (4.64), we obtain the epipolar line

$$\lambda_r \tilde{p}_r = \lambda_l A_r R A_l^{-1} \tilde{p}_l + A_r t , \tag{4.66}$$

where $A_r t$ is actually the epipole e_r in the second image, that is, the projection of the optical center C_l of the left camera into the right image.

By applying equation (4.66) to every image point in the left image, we can compute all the epipolar lines in the right image. The correspondence of one point in the left image will then need to be searched only along its corresponding epipolar line. Note that the epipolar lines pass all through the same epipole. However, observe that in computing equation (4.66) we did not take into account the radial distortion introduced by the lens. Although for some narrow-field-of-view cameras the radial distortion is rather small, it is always opportune to take the radial distortion into account when computing the equation of the epipolar line. The reason is that if the epipolar line is not determined precisely, the correspondence search along a non accurate epipolar line can lead to a larger uncertainty in the computation of the disparity as well as in the reconstruction of the scene point P_w.

Instead of taking into account the radial distortion, a common consolidated procedure in stereo vision is that of undistorting first the two images, that is, remapping the left and right image into new images without distortion. Furthermore, the two images can be remapped in such a way that all epipolar lines in the left and right image are collinear and horizontal (figure 4.47d). The process of transforming a pair of stereo images into a new pair without radial distortion and with horizontal epipolar lines is called *stereo rectification* or *epipolar rectification*. We will briefly explain it in the next section.

Epipolar rectification. Given a pair of stereo images, epipolar rectification is a transformation of each image plane such that all corresponding epipolar lines become collinear and parallel to one of the image axes, for convenience usually the horizontal axis. The resulting rectified images can be thought of as acquired by a new stereo camera obtained by rotating the original cameras about their optical centers. The great advantage of the epipolar recti-

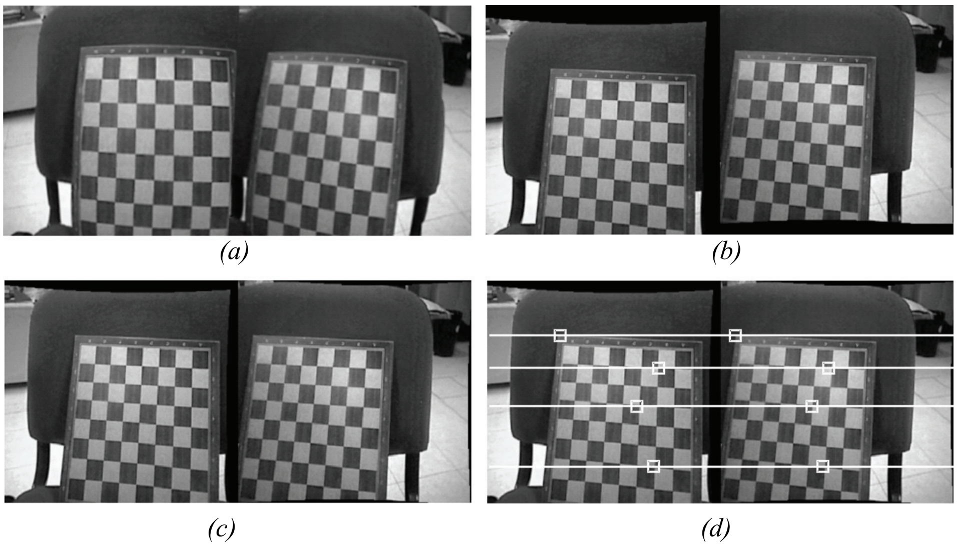


Figure 4.47 Rectification of a stereo pair: (a) original images, (b) compensation of the lens distortion, (c) compensation of rotation and translation, (d) After the epipolar rectification, the epipolar lines appear collinear and horizontal.

fication is the correspondence search becomes simpler and computationally less expensive because the search is done along the horizontal lines of the rectified images. The steps of the epipolar rectification algorithm are illustrated in figure 4.47. Observe that after the rectification, all the epipolar lines in the left and right image are collinear and horizontal (figure 4.47d). The equations for the epipolar rectification algorithm go beyond the scope of this book, but the interested reader can find an easy-to-implement algorithm in [133].

Disparity map. After the calibration of the stereo-rig, the epipolar rectification, and the correspondence search, we can finally reconstruct the scene points in 3D by solving the system of equations (4.60)–(4.61) (see also the problem in section 4.8). Another popular output of stereo vision is the *disparity map*. A disparity map appear as a grayscale image where the intensity of every pixel point is proportional to the disparity of that pixel in the left and right image: objects that are closer to the camera appear lighter, while farther objects appear darker. An example disparity map is shown in figure 4.48. Disparity maps are very useful for obstacle avoidance (figure 4.49). Modern stereo cameras—like those from Videre Design and Point-Grey (figure 4.42)—are able to compute disparity maps directly in hardware.

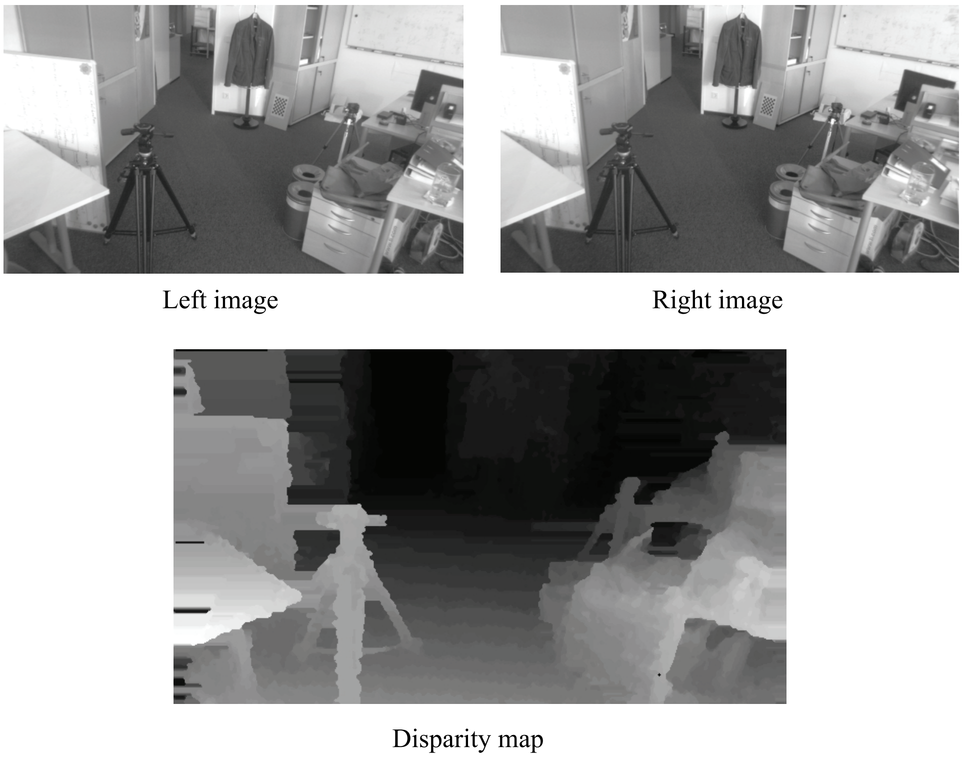


Figure 4.48 An example disparity map computed from the two top images. Every pixel point is proportional to the disparity of that pixel in the left and right image. Objects that are closer to the camera appear lighter, while farther objects appear darker. Image courtesy of Martin Humenberger, AIT Austrian Institute of Technology — http://www.ait.ac.at.

4.2.6 Structure from motion

In the previous section, we described how to recover the structure of the environment from two images of the scene taken from two distinct cameras whose relative position and orientation is known. In this section, we discuss the problem of recovering the structure when the camera relative pose is unknown. This is the case, for instance, when the two images are taken from the same camera but at different positions and at different times,[8] or, alternatively, from different cameras. This implies that both structure and motion must be estimated simultaneously. This problem is known as *Structure from Motion* (SfM). This problem has been studied for long time in the computer vision community, and in this sec-

8. For the sake of simplicity, here we assume that the scene is time invariant (i.e., static). One way to deal with dynamic scenes consists in treating moving objects as outliers.

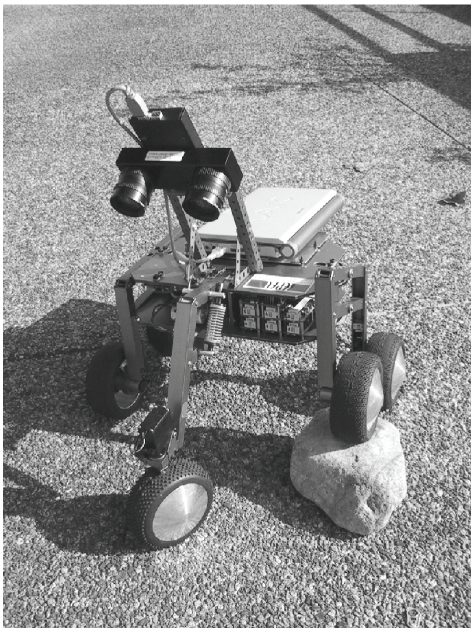


Figure 4.49
A stereo camera from Videre Design on the Shrimp robot developed at the ASL.

tion we provide only the solution to the two-frame structure from motion problem. For an in-depth study of structure from motion, we refer the reader to [21, 22, 29, 36, 53].

Observe that in structure-from-motion, the images do not need to be precalibrated. This allows SfM to work in challenging situations, where, for instance, images are taken by different users using different cameras (for example, images from the Web). The intrinsic parameters can in fact be estimated automatically from SfM itself. A suggestive result of SfM is illustrated in figure 4.50. Here, the scene was reconstructed using dozens of images. Using thousands of images from different viewpoints, SfM can sometimes achieve 3D reconstruction results that are almost comparable in accuracy and density of points to 3D laser rangefinders (page 133). However, this precision is often at the expense of the computation power.

4.2.6.1 Two-view structure-from-motion

Let us start again from the two perspective projection equations (4.60) and (4.61) derived for the stereo vision case, but now remember that R and t denote the relative motion between the first and the second camera position; therefore, we can write:

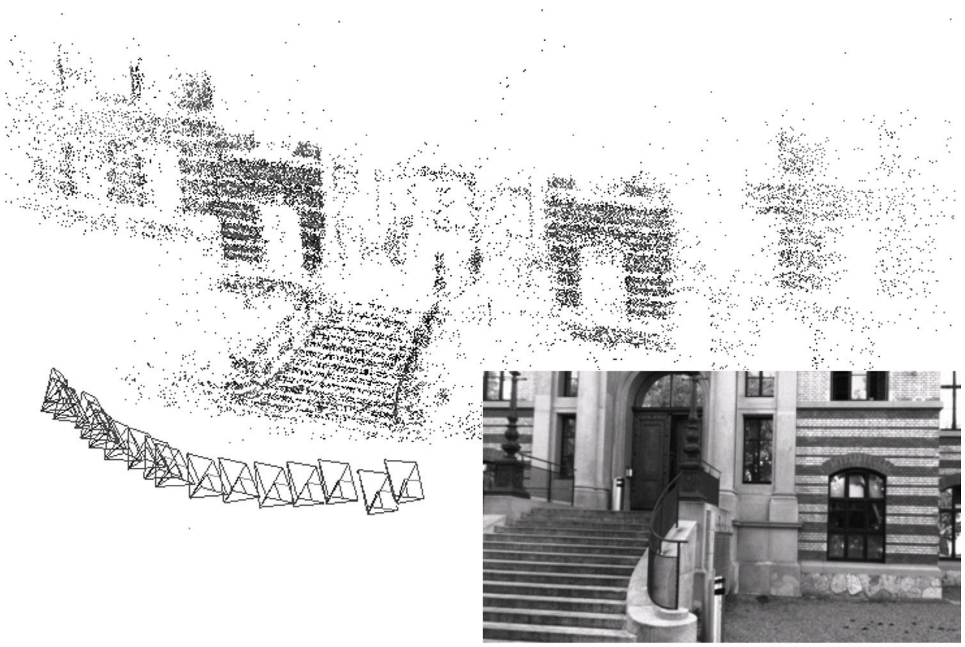


Figure 4.50 An example of structure-from-motion: salient image points (image features, see section 4.5) are extracted and matched across multiple frames. Wrong data associations (outliers) are removed and the relative motion among the views is determined. Finally, the points are reconstructed by triangulation. The reconstruction of the building was obtained using dozens of images. The camera poses are also displayed. Image courtesy of Friedrich Fraundorfer. Structure from motion also allows dense reconstruction of entire cities and monuments using just images.

$$\lambda_1 \tilde{p}_1 = A_1[I|0]\tilde{P}_w = A_1 P_w \text{ (for the first camera position)}, \tag{4.67}$$

$$\lambda_2 \tilde{p}_2 = A_2[R|t]\tilde{P}_w \text{ (for the second camera position)}. \tag{4.68}$$

In order to simplify our problem, let us make some assumptions. Let us assume that we use the same camera for the first and the second position and that the intrinsic parameters do not change in between; therefore, $A_1 = A_2 = A$. Let us also assume that the camera is calibrated, and that therefore A is known. In this case, it is more convenient to work with *normalized* image coordinates. Let $\tilde{x}_1$ and $\tilde{x}_2$ be the *normalized* coordinates of $\tilde{p}_1$ and $\tilde{p}_2$ respectively, where

$$\tilde{x}_1 = A^{-1}\tilde{p}_1 \text{ and } \tilde{x}_2 = A^{-1}\tilde{p}_2, \tag{4.69}$$

and $\tilde{x}_1 = (x_1, y_1, 1)$, $\tilde{x}_2 = (x_2, y_2, 1)$. Then, we can rewrite (4.67) and (4.68) as:

$$\lambda_1 \tilde{x}_1 = P_w \text{ (for the first camera position),} \tag{4.70}$$

$$\lambda_2 \tilde{x}_2 = [R|t]\tilde{P}_w = RP_w + t \text{ (for the second camera position).} \tag{4.71}$$

As we did before for computing the epipolar lines (page 176), let us map the optical ray corresponding to x_1 into the second image. Thus, by substituting (4.70) into (4.71), we obtain:

$$\lambda_2 \tilde{x}_2 = \lambda_1 R \tilde{x}_1 + t . \tag{4.72}$$

Let us now take the *cross* product of both sides with *t*. This in order to cancel *t* on the right-hand side. Then, we obtain:

$$\lambda_2 [t] \times \tilde{x}_2 = \lambda_1 ([t] \times R) \cdot \tilde{x}_1 , \tag{4.73}$$

where $[t]\times$ is an antisymmetric matrix defined as

$$[t] \times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} . \tag{4.74}$$

Now, taking the *dot* product of both sides (4.73) with x_2 yields:

$$\lambda_2 \tilde{x}_2^T \cdot ([t] \times \tilde{x}_2) = \lambda_1 \tilde{x}_2^T \cdot ([t] \times R) \cdot \tilde{x}_1 . \tag{4.75}$$

Observe that $\tilde{x}_2^T \cdot ([t] \times \tilde{x}_2) = 0$, and therefore from (4.75) we obtain:

$$\tilde{x}_2^T \cdot ([t] \times R) \cdot \tilde{x}_1 = 0 , \tag{4.76}$$

which is called *epipolar constraint*. Observe that the epipolar constraint is valid for every pair of conjugate points.

Let us define the *essential matrix* $E = ([t] \times R)$, the epipolar constraint reads as;

$$\tilde{x}_2^T \cdot E \cdot \tilde{x}_1 = 0 \tag{4.77}$$

It can be shown that the essential matrix has two singular values which are equal and another which is zero [29].

Computing the essential matrix. Given this fundamental relationship (4.77), how can we use it to recover the camera motion encoded in the essential matrix *E*? If we have *N* corresponding measurements $\{(x_1^i, x_2^i)\}$, we can form *N* homogeneous equations in the nine elements of $E = [e_{11}, e_{12}, e_{13}, e_{21}, e_{22}, e_{23}, e_{31}, e_{32}, e_{33}]^T$, of the type

$$\tilde{x}_1^i\tilde{x}_2^i e_{11} + \tilde{y}_1^i\tilde{x}_2^i e_{12} + \tilde{x}_2^i e_{13} + \tilde{x}_1^i\tilde{y}_2^i e_{21} + \tilde{y}_1^i\tilde{y}_2^i e_{22} + \tilde{y}_2^i e_{23} + \tilde{x}_1^i e_{31} + \tilde{y}_1^i e_{32} + e_{33} = 0 . \quad (4.78)$$

This can be rewritten in a more compact way as:

$$D \cdot E = 0 . \quad (4.79)$$

Given $N \geq 8$ such equations, we can compute an estimate (up to a scale) for the entries in *E* using the *Singular Values Decomposition* (*SVD*). The solution of (4.79) will therefore be the eigenvector of *D* corresponding to the smallest eigenvalue. Because at least eight point correspondences are needed, this algorithm is known as the *eight-point algorithm* [194]. This algorithm is one of the milestones of computer vision. The main advantages of the eight-point algorithm are that it is very easy to implement and that it works also for an uncalibrated camera, that is, when the camera intrinsic parameters are unknown. The drawback is that it does not work for degenerate point configurations such as planar scenes, that is, when all the scene points are coplanar.

In the case of a calibrated camera, at least five point correspondences are required [178]. An efficient algorithm for computing the essential matrix from at least five point correspondences was proposed by Nister [246]. The *five-point algorithm* works only for calibrated cameras but is more complicated to implement. However, in contrast to the eight-point algorithm, it also works for planar scenes.

Decomposing *E* into *R* and *t*. Let us now assume that the essential matrix *E* has been determined from known point correspondences. How do we determine *R* and *t*? Because a complete derivation of the proof is beyond the scope of this book, we will give directly the final expression. The interested reader can find the proof of these equations in [29].

Before decomposing *E*, we need to enforce the constraint that two of its singular values are equal and the third one is zero. In fact, in presence of image noise this constraint will never be verified in practice. To do this, we compute the closest[9] essential matrix $\hat{E}$ which

9. Closest in terms of the Frobenius norm.

satifies this constraint. One popular technique is to use SVD and force the two larger singular values to be equal and the smallest one to be zero. Therefore:

$$[U, S, V] = SVD(E), \tag{4.80}$$

where $S = diag([S_{11}\ S_{ss}\ \mathrm{S}_{33}])$ with $S_{11} \geq S_{22} \geq S_{33}$. Then, the closest essential matrix eo E in the Frobenius norm is given by

$$\hat{E} = U \cdot diag\left(\left[\frac{S_{11}+S_{22}}{2},\ \frac{S_{11}+S_{22}}{2},\ 0\right]\right) \cdot V^T \tag{4.81}$$

Then, we replace E with $\hat{E}$. At this point, we can decompose E into R and t.

The decomposition of E returns four solutions for (R,t), two for R and two for t. Let us define

$$B = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } [U, S, V] = SVD(E), \tag{4.82}$$

where U, S, and V are such that $U \cdot S \cdot V^T = E$. It can be shown (see [29]) that the two solutions for R are:

$$R_1 = det(U \cdot V^T) \cdot U \cdot B \cdot V^T, \tag{4.83}$$

$$R_2 = det(U \cdot V^T) \cdot U \cdot B^T \cdot V^T. \tag{4.84}$$

Now, let us define

$$L = U \cdot \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot U^T \text{ and } M = -U \cdot \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot U^T. \tag{4.85}$$

The two solutions for t are:

$$t_1 = \frac{\begin{bmatrix} L_{32} & L_{13} & L_{21} \end{bmatrix}^T}{\left\| \begin{bmatrix} L_{32} & L_{13} & L_{21} \end{bmatrix} \right\|}, \tag{4.86}$$

$$t_2 = \frac{\left[M_{32}\ M_{13}\ M_{21}\right]}{\left\| \left[M_{32}\ M_{13}\ M_{21}\right]^T \right\|}. \tag{4.87}$$

These four solutions can be disambiguated using the so-called *cheirality constraint*, which requires that reconstructed point correspondences lie in front of the cameras. In fact, if you analyse the four solutions of the SfM problem, you will always find that three solutions are such that the reconstructed point correspondences appear behind at least one of the two cameras, while only one solution guarantees that they lie in front of both cameras. Thus, testing with a single point correspondence to determine if it is reconstructed in front of both cameras is sufficient to identify the right solution out of the four possible choices. Also, observe that the solution for t is known up to a scale. In fact, with a single camera it is not possible to recover the absolute scale. For the same reason, the recovered structure will also be known up to a scale.

The last step in two-view structure-from-motion is the reconstruction of the scene. Once R and t have been found, the 3D structure can be computed via triangulation of the feature points as done for the stereo camera (page 173).

Free software for multi-view structure from motion. To conclude this section, we would like to point the reader to some interesting, free software to performe structure-from motion from unordered image collections. The most popular is Microsoft Photosynth (http://photosynth.net)—inspired by the research work on Photo Tourism [355]—which is based on the very popular open-source software Bundler (available at http://phototour.cs.washington.edu/bundler) and described in [297] and [298].

Very useful and fully open-source tools for on-line processing are: the Parallel Tracking and Mapping (PTAM) tool [358], the Vodoo camera tracker [356], and the ARToolkit [357].

Useful Matlab toolboxes for structure from motion are:

- FIT3D: http://www.fit3d.info
- Structure from Motion toolbox by V. Rabaud: http://code.google.com/p/vincents-structure-from-motion-matlab-toolbox
- Matlab Functions for Multiple View Geometry by A. Zissermann: http://www.robots.ox.ac.uk/~vgg/hzbook/code
- Structure and Motion Toolkit by P. Torr: http://cms.brookes.ac.uk/staff/PhilipTorr/Code/code_page_4.htm
- Matlab Code for Non-Rigid Structure from Motion using Factorisation by L. Torresani http://movement.stanford.edu/learning-nr-shape

Finally, see also the companies 2d3 (http://www.2d3.com) and Vicon (http://www.vicon.com).

4.2.6.2 Visual odometry

Directly linked to structure from motion is *visual odometry*. Visual odometry consists in estimating the motion of a robot or that of a vehicle by using visual input alone. The term "visual odometry" was coined in 2004 by Nister with his homonym landmark paper [245], where he showed successful results on different vehicles (on-road and off-road) using either a single camera or a stereo camera. The basic principle behind visual odometry is a simple iteration of two-view structure from motion that we have seen in the previous section.

Most of the work done about visual odometry has been produced using stereo cameras and can be traced back to 1980 with Moravec's work [236]. Similar work has also been reported elsewhere also (see [160, 174, 181, 244]). Furthermore, stereo visual odometry has also been successfully used on Mars by the NASA rovers since early 2004 [203]. Nevertheless, visual odometry methods for outdoor applications have also been produced, which use a single camera alone (see [107, 244, 278, 279, 307]).

The advantage of using a stereo camera compared to a single camera is that the measurements are directly provided in the absolute scale. Conversely, when using a single camera the absolute scale must be estimated in other ways (e.g., from knowledge of an element in the scene, or the distance between the camera and the ground plane) or using other sensors such as GPS, IMU, wheel odometry, or lasers.

Visual odometry aims at recovering only the trajectory of the vehicle. Nevertheless, it is not uncommon to see results showing also the 3D map of the environment which is usually a simple triangulation of the feature points from the estimated camera poses. An example visual odometry result using a single omnidirectional camera is shown in figure 4.51. Here the scale was obtained by exploiting the nonholonomic constraints of the vehicle as described in [277]. In this figure, visual odometry is performed over a 3 km trajectory. Notice the visible drift toward the end of the trajectory.

All visual odometry algorithms suffer from motion drift due to the integration of the relative displacements between consecutive poses which unavoidably accumulates errors over time. This drift becomes evident usually after a few hundred meters. But the results may vary depending on the abundance of features in the environment, the resolution of the cameras, the presence of moving objects like people or other passing vehicles, and the illumination conditions. Remember that motion drift is also present in wheel odometry, as will be described in section 5.2.4. However, the reason visual odometry is becoming more and more popular in both robotics and automotive is that drift can be canceled if the vehicle revisits a place that has already been observed previously. The possibility of performing location recognition (or place recognition) is one of the main advantages of vision com-

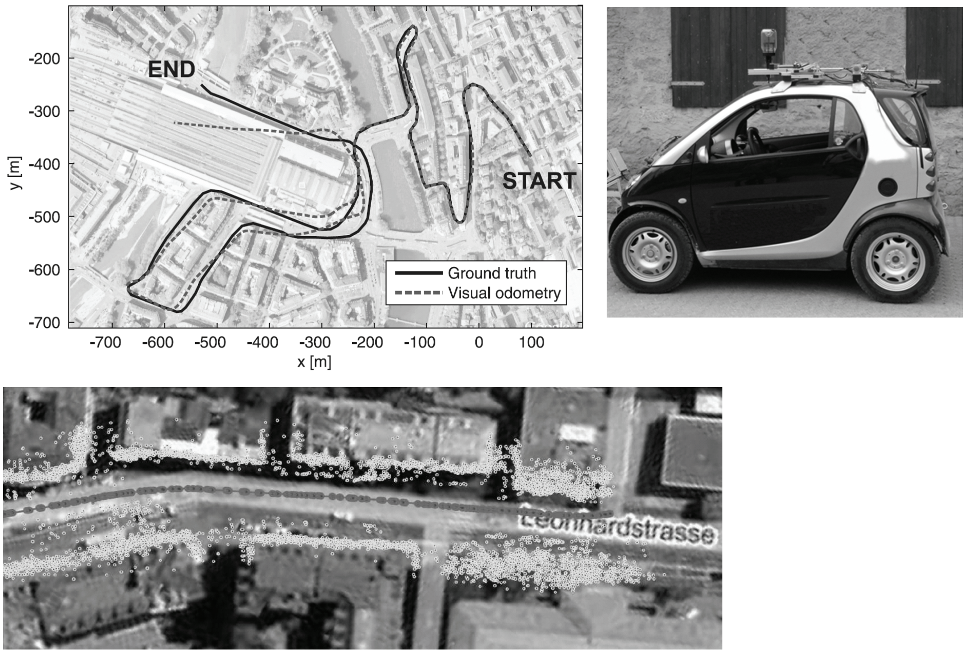


Figure 4.51 (upper left) An example visual odometry result with related map (bottom) obtained using a single omnidirectional camera mounted on the roof of the vehicle (right). The absolute scale was computed automatically by taking advantage of the fact that a wheeled vehicle is constrained to follow a circular coarse, locally, about the instantaneous center of rotation (see Ackerman steering principle in section 3.3.1). This visual odometry result was obtained using the 1-point RANSAC method described in [278]. The advantage of this algorithm compared to the state of the art is that visual odometry runs at 400 frames per second, while standard method work at 20–40 Hz.

pared to other sensor modalities. The most popular computer vision approaches for location recognition will be described in section 4.6. Once a place previously observed is visited a second time by the robot, the accumulated error can be reduced by adding the constraint that the positions of the vehicle at these two places (the previously visited and the revisited one) should actually coincide. This obviously requires an algorithm that modifies ("relaxes") all the previous robot poses until the error between the current and previously visited location is minimized.

The problem of location recognition is also called loop-detection, because a loop is a closed trajectory of a vehicle that returns to a previously-visited point. The problem of minimizing the error at the loop closure is instead called loop-closing. There are several algorithms in the literature to perform loop-closing. Some of them come from the computer vision community and rely on the so called *bundle adjustment*,[10] while others have been

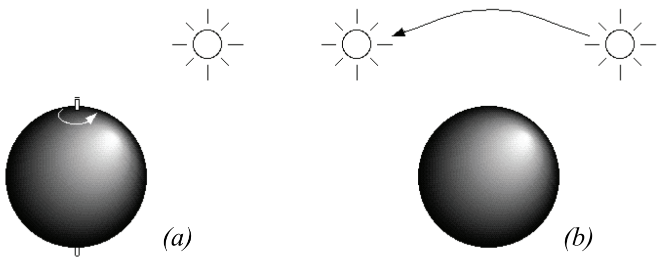


Figure 4.52
Motion of the sphere or the light source here demonstrates that optical flow is not always the same as the motion field.

developed within the robotics community to solve the Simultaneous Localization and Mapping (SLAM) problem (see section 5.8.2). Some of the most popular algorithms can be found in [318] and [352].

4.2.7 Motion and optical flow

A great deal of information can be recovered by recording time-varying images from a fixed (or moving) camera. First, we distinguish between the motion field and optical flow:

- Motion field: this assigns a velocity vector to every point in an image. If a point in the environment moves with velocity v_0, then this induces a velocity v_i in the image plane. It is possible to determine mathematically the relationship between v_i and v_0.
- Optical flow: it can also be true that brightness patterns in the image move as the object that causes them moves (light source). Optical flow is the apparent motion of these brightness patterns.

In our analysis here we assume that the optical flow pattern will correspond to the motion field, although this is not always true in practice. This is illustrated in figure 4.52a, where a sphere exhibits spatial variation of brightness, or shading, in the image of the sphere since its surface is curved. If the surface moves, however, this shading pattern will not move hence the optical flow is zero everywhere even though the motion field is not zero. In figure 4.52b, the opposite occurs. Here we have a fixed sphere with a moving light source. The shading in the image will change as the source moves. In this case the optical

10. Given a set of images observing a certain number of 3D points from different viewpoints, bundle adjustment is the problem of simultaneously refining the 3D coordinates of the scene geometry as well as the relative motion and the camera intrinsic parameters. This is done according to an optimality criterion involving the corresponding image projections of all points.

flow is nonzero but the motion field is zero. If the only information accessible to us is the optical flow and we depend on this, we will obtain incorrect results in both cases.

4.2.7.1 Optical flow

There are a number of techniques for attempting to measure optical flow and thereby obtain the scene's motion field. Most algorithms use local information, attempting to find the motion of a local patch in two consecutive images. In some cases, global information regarding smoothness and consistency can help to disambiguate further such *matching* processes. Below we present details for the optical flow constraint equation method. For more details on this and other methods, refer to [69, 151, 316].

Suppose first that the time interval between successive snapshots is so fast that we can assume that the measured intensity of a portion of the same object is effectively constant. Mathematically, let $I(x, y, t)$ be the image irradiance at time t at the image point (x, y). If $u(x, y)$ and $v(x, y)$ are the x and y components of the optical flow vector at that point, we need to search a new image for a point where the irradiance will be the same at time $t + \delta t$, that is, at point $(x + \delta t, y + \delta t)$, where $\delta x = u\delta t$ and $\delta y = v\delta t$. That is,

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \tag{4.88}$$

for a small time interval, δt. This will capture the motion of a constant-intensity *patch* through time. If we further assume that the brightness of the image varies smoothly, then we can expand the left-hand side of equation (4.88) as a Taylor series to obtain

$$I(x, y, t) + \delta x\frac{\partial I}{\partial x} + \delta y\frac{\partial I}{\partial y} + \delta t\frac{\partial I}{\partial t} + e = I(x, y, t), \tag{4.89}$$

where e contains second- and higher-order terms in δx, and so on. In the limit as δt tends to zero we obtain

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0, \tag{4.90}$$

from which we can abbreviate

$$u = \frac{dx}{dt}\ ;\ \ v = \frac{dy}{dt} \tag{4.91}$$

and

$$I_x = \frac{\partial I}{\partial x} ; \quad I_y = \frac{\partial I}{\partial y} ; \quad I_t = \frac{\partial I}{\partial t} = 0 , \tag{4.92}$$

so that we obtain

$$I_x u + I_y v + I_t = 0 . \tag{4.93}$$

The derivative I_t represents how quickly the intensity changes with time while the derivatives I_x and I_y represent the spatial rates of intensity change (how quickly intensity changes across the image). Altogether, equation (4.93) is known as the *optical flow constraint equation*, and the three derivatives can be estimated for each pixel given successive images.

We need to calculate both *u* and *v* for each pixel, but the optical flow constraint equation only provides one equation per pixel, and so this is insufficient. The ambiguity is intuitively clear when one considers that a number of equal-intensity pixels can be inherently ambiguous—it may be unclear which pixel is the resulting location for an equal-intensity originating pixel in the prior image.

The solution to this ambiguity requires an additional constraint. We assume that in general the motion of adjacent pixels will be similar, and that therefore the overall optical flow of all pixels will be smooth. This constraint is interesting in that we know it will be violated to *some* degree, but we enforce the constraint nonetheless in order to make the optical flow computationally tractable. Specifically, this constraint will be violated precisely when different objects in the scene are moving in different directions with respect to the vision system. Of course, such situations will tend to include edges, and so this may introduce a useful visual cue.

Because we know that this smoothness constraint will be somewhat incorrect, we can mathematically define the degree to which we violate this constraint by evaluating the formula

$$e_s = \iint (u^2 + v^2) dx dy , \tag{4.94}$$

which is the integral of the square of the magnitude of the gradient of the optical flow. We also determine the error in the optical flow constraint equation (which in practice will not quite be zero).

$$e_c = \iint (I_x u + I_y v + I_t)^2 dx dy . \tag{4.95}$$

Both of these equations should be as small as possible, so we want to minimize $e_s + \lambda e_c$, where λ is a parameter that weights the error in the image motion equation relative to the departure from smoothness. A large parameter should be used if the brightness measurements are accurate and small if they are noisy. In practice, the parameter λ is adjusted manually and interactively to achieve the best performance.

The resulting problem then amounts to the calculus of variations, and the Euler equations yield

$$\nabla^2 u = \lambda(I_x u + I_y v + I_t)I_x, \tag{4.96}$$

$$\nabla^2 v = \lambda(I_x u + I_y v + I_t)I_y, \tag{4.97}$$

where

$$\nabla^2 = \frac{\partial^2}{\delta x^2} + \frac{\partial^2}{\delta y^2}, \tag{4.98}$$

which is the Laplacian operator.

Equations (4.96) and (4.97) form a pair of elliptical second-order partial differential equations that can be solved iteratively.

Where occlusions (one object occluding another) occur, discontinuities in the optical flow will occur. This, of course, violates the smoothness constraint. One possibility is to try to find edges that are indicative of such occlusions, excluding the pixels near such edges from the optical flow computation so that smoothness is a more realistic assumption. Another possibility is to make use of these distinctive edges opportunistically. In fact, corners can be especially easy to *pattern-match* across subsequent images and thus can serve as fiducial markers for optical flow computation in their own right.

Optical flow is an important ingredient in vision algorithms that combine cues across multiple algorithms. Obstacle avoidance and navigation control systems for mobile robots (especially flying robots) using optical flow have proved to be broadly effective as long as texture is present [23, 54].

4.2.8 Color tracking

An important aspect of vision sensing is that the vision chip can provide sensing modalities and cues that no other mobile robot sensor provides. One such novel sensing modality is detecting and tracking color in the environment.

Color is an environmental characteristic and represents both a natural cue and an artificial cue that can provide new information to a mobile robot. For example, the annual robot

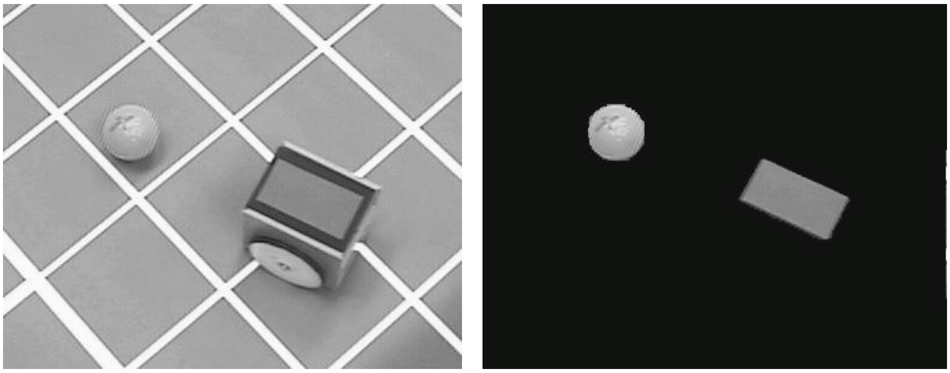


Figure 4.53
Color markers on the top of EPFL's STeam Engine soccer robots enable a color-tracking sensor to locate the robots and the ball in the soccer field.

soccer events (RoboCup) make extensive use of color both for environmental marking and for robot localization (see figure 4.53).

Color sensing has two important advantages. First, detection of color is a straightforward function of a single image, therefore no correspondence problem needs be solved in such algorithms. Second, because color sensing provides a new, independent environmental cue, if it is combined (i.e., *sensor fusion*) with existing cues, such as data from stereo vision or laser rangefinding, we can expect significant information gains.

Efficient color-tracking sensors are also available commercially—such as the CMUcam from Carnegie Mellon University—but they can also be implemented straightforwardly using a standard camera. The simplest way of doing this is using *constant thresholding*: a given pixel point is selected if and only if its RGB values (r, g, b) fall simultaneously in the chosen R, G, and B ranges, which are defined by six thresholds $[R_{min}, R_{max}]$, $[G_{min}, G_{max}]$, $[B_{min}, B_{max}]$. Therefore

$$R_{min} < r < R_{max} \text{ and } G_{min} < g < G_{max} \text{ and } B_{min} < b < B_{max} . \tag{4.99}$$

If we represent the RGB color space as a three-dimensional Euclidean space, the aforementioned method selects those pixels whose color components belong to the cube specified by the given thresholds. Alternatively, a sphere could be used. In this case, a pixel would be selected only if its RGB components are within a certain distance from a given point in the RGB space.

Alternatively to RGB, the YUV color space can be used. While R, G, and B values encode the intensity of each color, YUV separates the color (or *chrominance*) measure

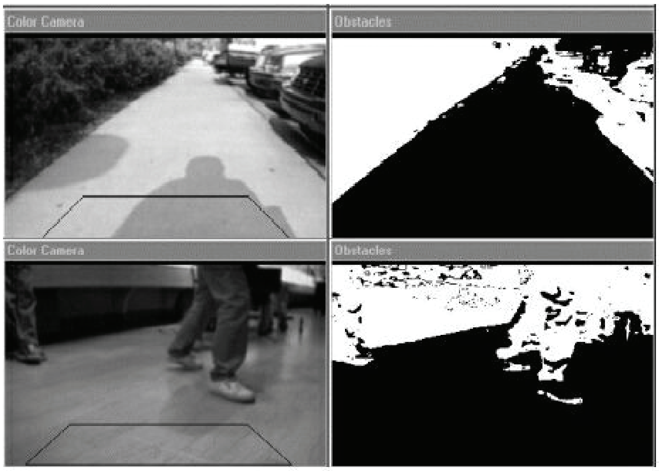


Figure 4.54
Examples of adaptive floor plane extraction. The trapezoidal polygon identifies the floor sampling region.

from the brightness (or *luminosity*) measure. Y represents the image's luminosity while U and V together capture its chrominance. Thus, a bounding box expressed in YUV space can achieve greater stability with respect to changes in illumination than is possible in RGB space.

A popular application of color segmentation in robotics is floor plane extraction (figure 4.54). In this case, color segmentation techniques more complex than color thresholding are used, like adaptive thresholding, or *k-means clustering*[11] [18]. Floor plane extraction is a vision approach for identifying the traversable portions of the ground. Because it makes use of edges (section 4.3.2) and color in a variety of implementations, such obstacle detection systems can easily detect obstacles in cases that are difficult for traditional ranging devices. As is the case with all vision algorithms, floor plane extraction succeeds only in environments that satisfy several important assumptions:

- Obstacles differ in appearance from the ground.
- The ground is flat, and its angle to the camera is known.

11. In statistics and machine learning, *k-means clustering* is a method of cluster analysis that aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. The k-means clustering algorithm is commonly used in computer vision as a form of image segmentation.

- There are no overhanging obstacles.

The first assumption is a requirement in order to discriminate the ground from obstacles using its appearance. A stronger version of this assumption, sometimes invoked, states that the ground is uniform in appearance and different from all obstacles. The second and third assumptions allow floor-plane-extraction algorithms to estimate the robot's distance to obstacles detected.

4.3 Fundamentals of Image Processing

Image processing is a form of signal processing where the input signal is an image (such as a photo or a video) and the output is either an image or a set of parameters associated with the image. Most image-processing techniques treat the image as a two-dimensional signal $I(x, y)$ where x and y are the *spatial* image coordinates and the amplitude of I at any pair of coordinates (x, y) is called *intensity* or *gray level* of the image at that point.

Image processing is a huge field and typical operations, among many others, are:

- Filtering, image enhancing, edge detection
- Image restoration and reconstruction
- Wavelets and multiresolution processing
- Image compression (e.g., JPEG)
- Euclidean geometry transformations such as enlargement, reduction, and rotation
- Color corrections such as brightness and contrast adjustments, quantization, or color translation to a different color space
- Image registration (the alignment of two or more images)
- Image recognition (for example, extracting a face from the image by using some face recognition algorithm)
- Image segmentation (partitioning the image in characteristic regions according to color, edges, or other features)

Because a review of all these techniques goes beyond the scope of this book, here we focus only on the most important image processing operations that are relevant for robotics. In particular, we describe image filtering operations such as smoothing and edge detection. We will then describe some image similarity measures for finding point correspondences between images, which are helpful in structure from stereo and structure from motion. For an in-depth study of image processing in general, we refer the reader to [26].

4.3.1 Image filtering

Image filtering is one of the principal tools in image processing. The word *filter* comes from frequency domain processing, where "filtering" refers to the process of accepting or rejecting certain frequency components. For example, a filter that passes low frequencies is called a *lowpass* filter. The effect produced by a lowpass filter is to blur (smooth) an image, which has the main effect of reducing image noise. Conversely, a filter that passes high frequencies is called *highpass* filter and is typically used for edge detection. Image filters can be implemented both in the frequency domain and in the spatial domain. In the latter case, the filter is called *mask* or *kernel*. In this section, we will review the fundamentals of spatial filtering.

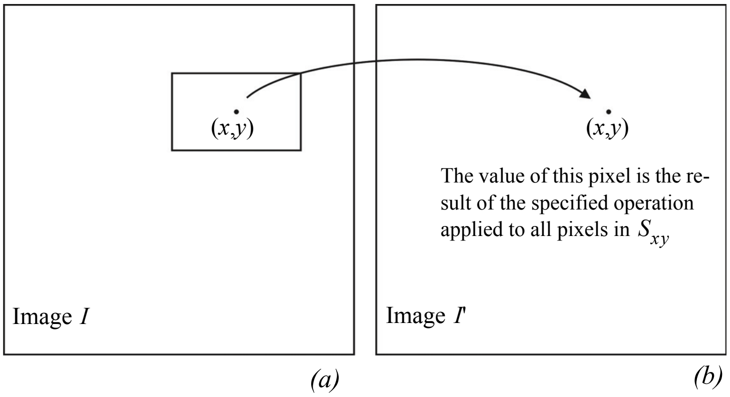
In figure 4.55, the basic principle of spatial filtering is explained. A spatial filter consists of (1) a neighborhood of the pixel under examination, (typically a small rectangle), and (2) a predefined operation T that is performed on the image pixels encompassed by the neighborhood. Let S_{xy} denote the set of coordinates of a neighborhood centered on an arbitrary point (x,y) in an image I. Spatial filtering generates a corresponding pixel at the same coordinates in an output image I' where the value of that pixel is determined by a specified operation on the pixels in S_{xy}. For example, suppose that the specified operation is to compute the average value of the pixels in a rectangular window of size $m \times n$ centered on (x,y). The locations of pixels in this region constitute the set S_{xy}. Figure 4.55a–b illustrates the process. We can express this operation in equation form as

$$I'(x, y) = \frac{1}{mn} \sum_{(r, c) \in S_{xy}} I(r, c) \tag{4.100}$$

where r and c are the row and column coordinates of the pixels in the set S_{xy}. The new image I' is created by varying the coordinates (x, y) so that the center of the window moves from pixel to pixel in image I. For instance the image in figure 4.55d was created in this manner using a window of size 21×21 applied on the image in figure 4.55c.

The filter used to illustrate the example above is called *averaging filter*. More generally, the operation performed on the image pixels can be linear or nonlinear. In these cases, the filter is called either a *linear* or *nonlinear* filter. Here, we concentrate on linear filters. In general, linear spatial filtering of an image with a filter w of size $m \times n$ is given by the expression

$$I'(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) \cdot I(x+s, y+t), \tag{4.101}$$


(x,y)
(x,y)
The value of this pixel is the result of the specified operation applied to all pixels in S_{xy}
Image I
Image I'
(a)
(b)

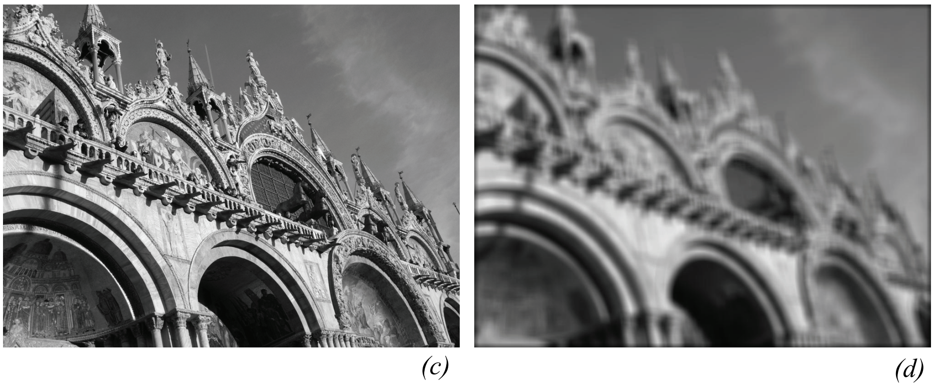


Figure 4.55 Illustration of the concept of spatial filtering. (c) Input image. (d) Output image after application of average filter.

where $m = 2a+1$ and $n = 2b+1$ are usually assumed odd integers. The filter w is also called *kernel*, *mask*, or *window*. As observed in (4.101), linear filtering is the process of moving a filter mask over the entire image and computing the sum of products at each location. In signal processing, this particular operation is also called *correlation* with the kernel w. It is, however, opportune to specify that an equivalent linear filtering operation is the *convolution*

$$I'(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot I(x-s, y-t) \tag{4.102}$$

where the only difference with the correlation is the presence of the minus sign, meaning that the image must be flipped. Observe that for symmetric filters *convolution* and *correlation* return the same result and the two terms can therefore be used interchangeably. The operation of convolution with the kernel w can be written in a more compact way as

$$I'(x, y) = w(x, y) * I(x, y), \tag{4.103}$$

where $*$ denotes the convolution operator.

Generating linear spatial filters requires that we specify the mn coefficients of the kernel. These coefficients are chosen based on what the filter is supposed to do. In the next section, we will see how to select these coefficients.

4.3.1.1 Smoothing filters

Smoothing filters are used for blurring and for noise reduction. Blurring is used in tasks such as removal of small details or filling of small gaps in lines or curves. Both blurring and noise reduction can be accomplished via linear or nonlinear filters. Here, we review some linear filters.

The output of a smoothing filter is simply the weighted average of the pixels contained in the filter mask. These filters are sometimes called *averaging filters* or *lowpass filters*. As we explained before, every pixel in an image is replaced by the average of the intensity of the pixels in the neighborhood defined by the filter mask. This process results in a new image with reduced sharp transitions. Accordingly, image noise gets reduced. As a side effect, however, edges—which are usually a desirable feature of an image—also get blurred. This side effect can be limited by choosing the filter coefficients appropriately. Finally observe that a *nonlinear* averaging filter can also be easily implemented by taking the *median* of the pixels contained in the mask. Median filters are particularly useful to remove *salt and pepper* noise.[12]

In the previous section, we already saw an example of constant averaging filter (figure 4.55) that simply yields the standard average of the pixels in the mask. Assuming a 3×3 mask, the filter can be written as

$$w = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \tag{4.104}$$

12. Salt and pepper noise represents itself as randomly occurring white and black pixels and is a typical form of noise in images.

where all the coefficients sum to 1. This normalization is important to keep the same value as the original image if the region by which the filter is multiplied is uniform. Also note that, instead of being 1/9, the coefficients of the filter are all 1s. The idea is that the pixels are first summed up and the result is then divided by 9. Indeed, this is computationally more efficient than multiplying each element by 1/9.

Many image-processing algorithms make use of the second derivative of the image intensity. Because of the susceptibility of such high-order derivative algorithms to changes in illumination in the basic signal, it is important to smooth the signal so that changes in intensity are due to real changes in the luminosity of objects in the scene rather than random variations due to imaging noise. A standard approach is the use of a Gaussian averaging filter whose coefficients are given by

$$G_{\sigma}(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} . \tag{4.105}$$

To generate, say, a 3×3 filter mask from this function, we sample it about its center. For example, with $\sigma = 0.85$, we get

$$G = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \tag{4.106}$$

where, again, the coefficients were rescaled so that they sum to 1. Also notice that the coefficients are all powers of 2, which makes it extremely efficient to compute. This filter is actually very popular. Such a lowpass filter effectively removes high-frequency noise, and this in turn causes the first derivative and especially the second derivative of intensity to be far more stable. Because of the importance of gradients and derivatives to image processing, such Gaussian smoothing preprocessing is a popular first step of virtually all computer vision algorithms.

4.3.2 Edge detection

Figure 4.56 shows an image of a scene containing a part of a ceiling lamp as well as the edges extracted from this image. Edges define regions in the image plane where a *significant* change in the image brightness takes place. As shown in this example, edge detection significantly reduces the amount of information in an image, and is therefore a useful potential feature during image interpretation. The hypothesis is that edge contours in an image correspond to important scene contours. As figure 4.56b shows, this is not entirely true. There is a difference between the output of an edge detector and an ideal line drawing.

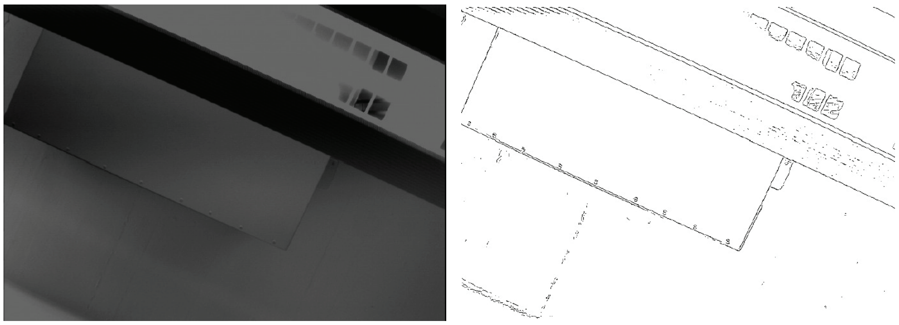

(a) *(b)*

Figure 4.56
(a) Photo of a ceiling lamp. (b) Edges computed from (a).

Typically, there are missing contours, as well as noise contours, that do not correspond to anything of significance in the scene.

The basic challenge of edge detection is visualized in figure 4.57. The top left portion shows the 1D section of an ideal edge. But the signal produced by a camera will look more like figure 4.57 (top right) because of noise. The location of the edge is still at the same x value, but a significant level of high-frequency noise affects the signal quality.

A naive edge detector would simply differentiate, since an edge by definition is located where there are large transitions in intensity. As shown in figure 4.57 (bottom right), differentiation of the noisy camera signal results in subsidiary peaks that can make edge detection very challenging. A far more stable derivative signal can be generated simply by preprocessing the camera signal using the Gaussian smoothing function described above. Below, we present several popular edge detection algorithms, all of which operate on this same basic principle, that the derivative(s) of intensity, following some form of smoothing, comprises the basic signal from which to extract edge features.

Optimal edge detection: the Canny edge detector. The current reference edge detector throughout the vision community was invented by John Canny in 1983 [91]. This edge detector was born out of a formal approach in which Canny treated edge detection as a signal-processing problem in which there are three explicit goals:

- Maximizing the signal-to-noise ratio;
- Achieving the highest precision possible on the location of edges;
- Minimizing the number of edge responses associated with each edge.

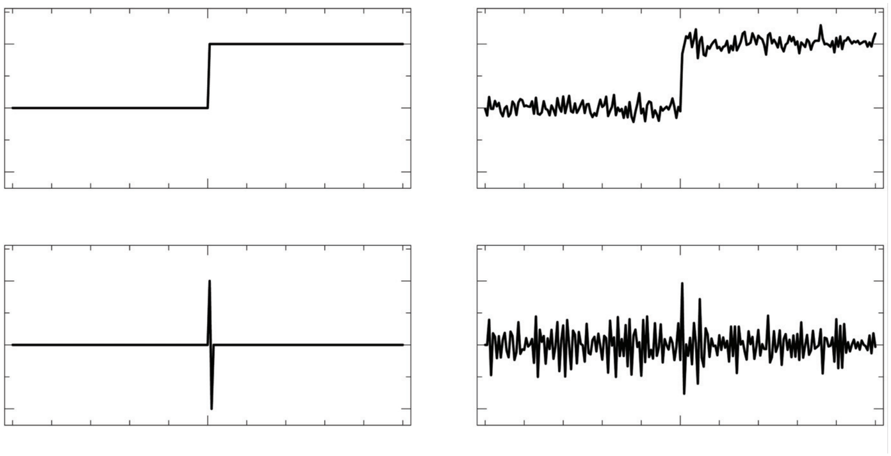


Figure 4.57
Step function example of second derivative shape and the impact of noise.

The Canny edge extractor smooths the image *I* via Gaussian convolution and then looks for maxima in the (rectified) derivative. In practice, the smoothing and differentiation are combined into one operation because

$$(G*I)' = G'*I \quad .^{13} \tag{4.107}$$

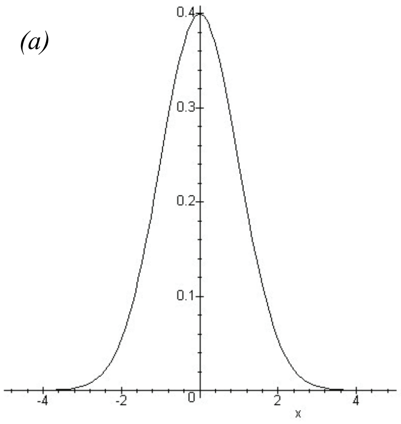
Thus, smoothing the image by convolving with a Gaussian G_σ and then differentiating is equivalent to convolving the image with G'_σ, which is the first derivative of G_σ (figure 4.58b).

We wish to detect edges in any direction. Since G' is directional, this requires application of two perpendicular filters (figure 4.59). We define the two filters as $f_V(x, y) = G'_\sigma(x)G_\sigma(y)$ and $f_H(x, y) = G'_\sigma(y)G_\sigma(x)$. The result is a basic algorithm for detecting edges at arbitrary orientations:

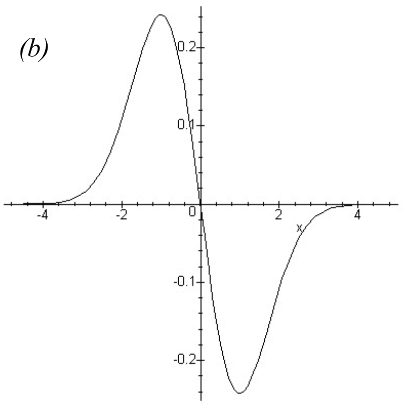
The algorithm for detecting edge pixels at an arbitrary orientation is as follows:

1. Convolve the image $I(x, y)$ with $f_V(x, y)$ and $f_H(x, y)$ to obtain the gradient components $R_V(x, y)$ and $R_H(x, y)$, respectively.
2. Define the square of the gradient magnitude $R(x, y) = R_V^2(x, y) + R_H^2(x, y)$.
3. Mark those peaks in $R(x, y)$ that are above some predefined threshold T.

13. This is a known property of convolution.

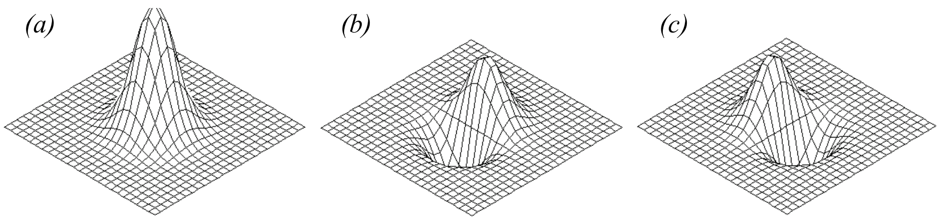


$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$



$$G_\sigma'(x) = \frac{-x}{\sigma^3\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

Figure 4.58
(a) A Gaussian function. (b) The first derivative of a Gaussian function.



$G_\sigma(x, y) = G_\sigma(x)G_\sigma(y)$ $f_V(x, y) = G'_\sigma(x)G_\sigma(y)$ $f_H(x, y) = G'_\sigma(y)G_\sigma(x)$

Figure 4.59
(a) Two-dimensional Gaussian function. (b) Vertical filter. (c) Horizontal filter.

Once edge pixels are extracted, the next step is to construct complete edges. A popular next step in this process is *nonmaxima suppression*. Using edge direction information, the process involves revisiting the gradient value and determining whether or not it is at a local maximum. If not, then the value is set to zero. This causes only the maxima to be preserved, and thus reduces the thickness of all edges to a single pixel (figure 4.60).

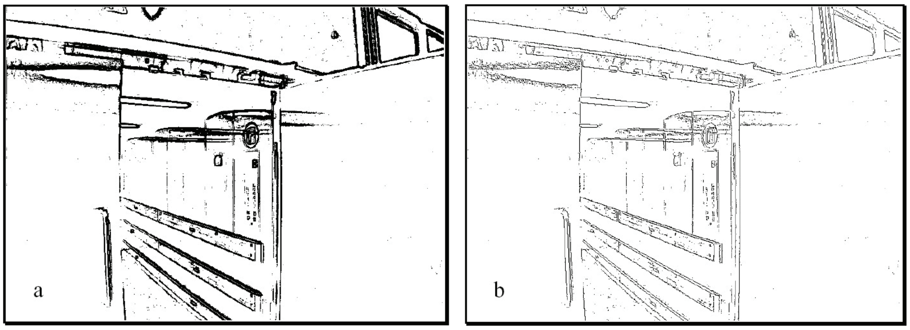


Figure 4.60
(a) Example of an edge image; (b) Nonmaxima suppression of (a).

Finally, we are ready to go from edge pixels to complete edges. First, find adjacent (or connected) sets of edges and group them into ordered lists. Second, use thresholding to eliminate the weakest edges.

Gradient edge detectors. On a mobile robot, computation time must be minimized to retain the real-time behavior of the robot. Therefore simpler, discrete kernel operators are commonly used to approximate the behavior of the Canny edge detector. One such early operator was developed by Roberts in 1965 [43]. He used two 2×2 masks to calculate the gradient across the edge in two diagonal directions. Let r_1 be the value calculated from the first mask and r_2 that from the second mask. Roberts obtained the gradient magnitude $|G|$ with the equation

$$|G| \cong \sqrt{r_1^2 + r_2^2} \;;\quad r_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \;;\quad r_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{4.108}$$

Prewitt (1970) [43] used two 3×3 masks oriented in the row and column directions. Let p_1 be the value calculated from the first mask and p_2 the value calculated from the second mask. Prewitt obtained the gradient magnitude $|G|$ and the gradient direction θ taken in a clockwise angle with respect to the column axis shown in the following equation.

$$|G| \cong \sqrt{p_1^2 + p_2^2} \;;$$

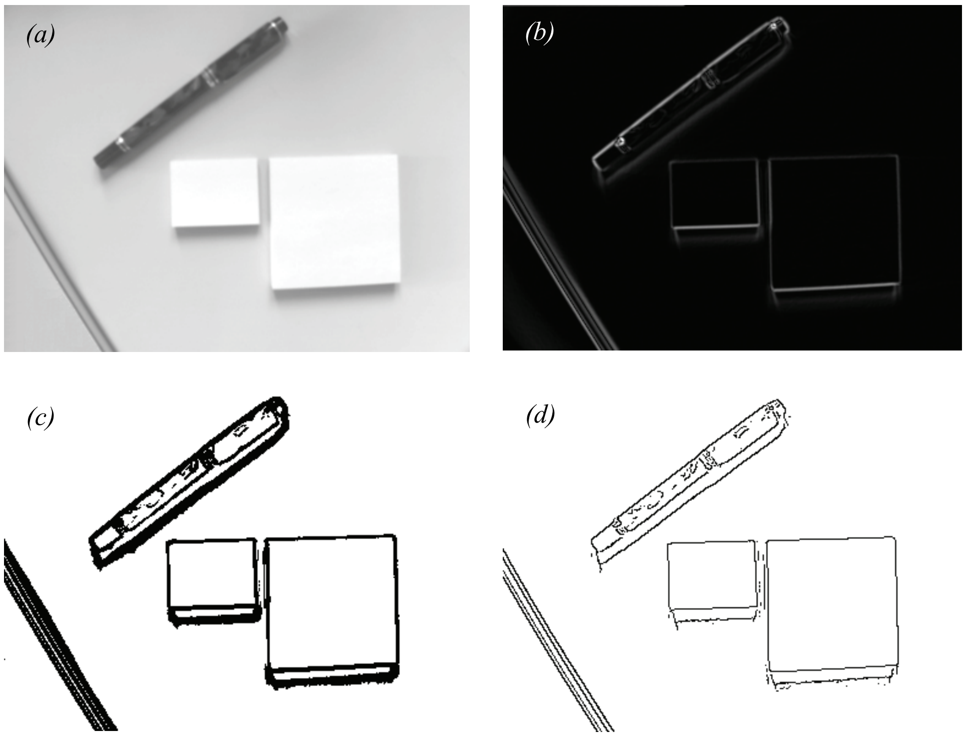


Figure 4.61
Example of visual feature extraction with the different processing steps: (a) raw image data; (b) filtered image using a Sobel filter; (c) thresholding, selection of edge pixels (d) nonmaxima suppression.

$$\theta \cong \operatorname{atan}\left(\frac{p_1}{p_2}\right) \quad ; \quad p_1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad ; \quad p_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.109}$$

In the same year, Sobel [43] used, like Prewitt, two 3×3 masks oriented in the row and column direction. Let s_1 be the value calculated from the first mask and s_2 the value calculated from the second mask. Sobel obtained the same results as Prewitt for the gradient magnitude $|G|$ and the gradient direction θ taken in a clockwise angle with respect to the column axis. Figure 4.61 shows application of the Sobel filter to a visual scene.

$$|G| \cong \sqrt{s_1^2 + s_2^2} \ ;$$

$$\theta \cong \operatorname{atan}\left(\frac{s_1}{s_2}\right) \ ; \qquad s_1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \ ; \ s_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} . \tag{4.110}$$

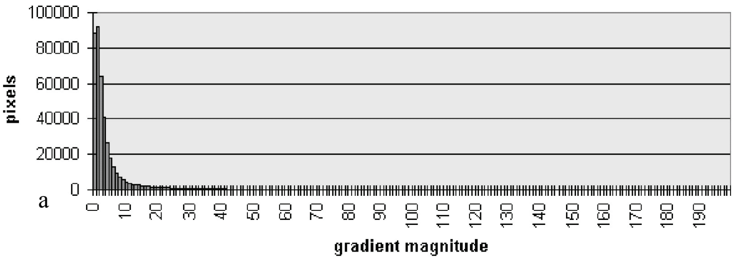
Dynamic thresholding. Many image-processing algorithms have generally been tested in laboratory conditions or by using static image databases. Mobile robots, however, operate in dynamic real-world settings where there is no guarantee regarding optimal or even stable illumination. A vision system for mobile robots has to adapt to the changing illumination. Therefore a constant threshold level for edge detection is not suitable. The same scene with different illumination results in edge images with considerable differences. To adapt the edge detector dynamically to the ambient light, a more adaptive threshold is required, and one approach involves calculating that threshold based on a statistical analysis of the image about to be processed.

To do this, a histogram of the gradient magnitudes of the processed image is calculated (figure 4.62). With this simple histogram it is easy to consider only the n pixels with the highest gradient magnitude for further calculation steps. The pixels are counted backward, starting at the highest magnitude. The gradient magnitude of the point where n is reached will be used as the temporary threshold value.

The motivation for this technique is that the n pixels with the highest gradient are expected to be the most relevant ones for the processed image. Furthermore, for each image, the same number of relevant edge pixels is considered, independent of illumination. It is important to pay attention to the fact that the number of pixels in the edge image delivered by the edge detector is not n. Because most detectors use nonmaxima suppression, the number of edge pixels will be further reduced.

Straight edge extraction: Hough transform. In mobile robotics, the straight edge is often extracted as a specific feature. Straight vertical edges, for example, can be used as clues to the location of doorways and hallway intersections. The Hough transform is a simple tool for extracting edges of a particular shape [21, 28]. Here we explain its application to the problem of extracting straight edges.

Suppose a pixel (x_p, y_p) in the image I is part of an edge. Any straight-line edge including point (x_p, y_p) must satisfy the equation: $y_p = m_1 x_p + b_1$. This equation can only be satisfied with a constrained set of possible values for m_1 and b_1. In other words, this equation is satisfied only by lines through I that pass through (x_p, y_p).

a


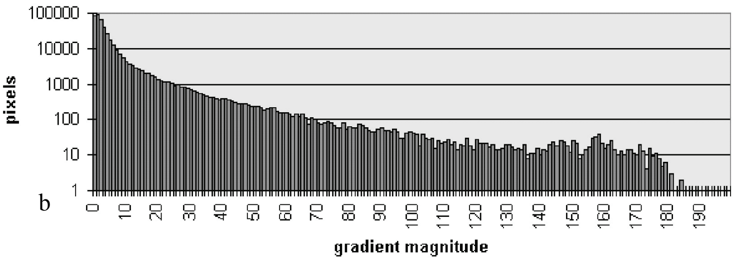
b


Figure 4.62
(a) Number of pixels with a specific gradient magnitude in the image of figure 4.61b. (b) Same as (a), but with logarithmic scale

Now consider a second pixel, (x_q, y_q) in I. Any line passing through this second pixel must satisfy the equation: $y_q = m_2x_q + b_2$. What if $m_1 = m_2$ and $b_1 = b_2$? Then the line defined by both equations is one and the same: it is the line that passes through both (x_p, y_p) and (x_q, y_q).

More generally, for all pixels that are part of a single straight line through I, they must all lie on a line defined by the *same* values for m and b. The general definition of this line is, of course, $y = mx + b$. The Hough transform uses this basic property, creating a mechanism so that each edge pixel can "vote" for various values of the (m, b) parameters. The lines with the most votes at the end are straight edge features:

- Create a 2D array *A* with axes that tessellate the values of *m* and *b.*
- Initialize the array to zero: $A[m, b] = 0$ for all values of m, b.
- For each edge pixel (x_p, y_p) in I, loop over all values of m and b:
  if $y_p = mx_p + b$ then $A[m, b]$+=1 .
- Search the cells in A to identify those with the largest value. Each such cell's indices (m, b) correspond to an extracted straight-line edge in I.

4.3.3 Computing image similarity

In this section, we review the three most popular image similarity measures used for solving the correspondence problem in structure from stereo (section 4.2.5) and structure from motion (section 4.2.6). The methods we are about to describe are all *area-based* (page 174). Suppose that we want to compare a $m \times n$ patch in image I_1 centered on (u,v) with another patch of the same size centered on (u', v') in image I_2. We assume that these are odd integers, therefore $m = 2a + 1$ and $n = 2b + 1$. The similarity is then computed between the gray intensity levels of the two patches. Some of the most popular criteria are:

Sum of Absolute Differences (SAD)

$$SAD = \sum_{k=-a}^{a} \sum_{l=-b}^{b} \left| I_1(u+k, v+l) - I_2(u'+k, v'+l) \right| . \tag{4.111}$$

Sum of Squared Differences (SSD)

$$SSD = \sum_{k=-a}^{a} \sum_{l=-b}^{b} [I_1(u+k, v+l) - I_2(u'+k, v'+l)]^2 . \tag{4.112}$$

Normalized Cross Correlation (NCC)

$$NCC = \frac{\sum_{k=-a}^{a} \sum_{l=-b}^{b} [I_1(u+k, v+l) - \mu_1] \cdot [I_2(u'+k, v'+l) - \mu_2]}{\sqrt[2]{\sum_{k=-a}^{a} \sum_{l=-b}^{b} [I_1(u+k, v+l) - \mu_1]^2 \sum_{k=-a}^{a} \sum_{l=-b}^{b} [I_2(u'+k, v'+l) - \mu_2]^2}} , \tag{4.113}$$

where

$$\mu_1 = \frac{1}{mn} \sum_{k=-a}^{a} \sum_{l=-b}^{b} I_1(u+k, v+l) \tag{4.114}$$

$$\mu_2 = \frac{1}{mn} \sum_{k=-a}^{a} \sum_{l=-b}^{b} I_2(u'+k, v'+l) \tag{4.115}$$

are the mean values of the two image patches.

The SAD is the simplest among these similarity measures. It is calculated by subtracting pixels between the reference image I_1 and the target image I_2 followed by the aggregation of absolute differences within the patch. The SSD has a higher computational complexity compared to the SAD, since it involves numerous multiplication operations (i.e., squared). Notice that if the left and right images match perfectly, the resultant of SAD and SSD will be zero.

The NCC is even more complex than both SAD and SSD algorithms, since it involves numerous multiplication, division, and square root operations; however, it but provides more distinctiveness than SSD and SAD and also invariance to affine intensity changes (see also figure 4.69). Finally, note that the value of NCC ranges between -1 and 1 where 1 corresponds to maximum similarity between the two image patches.

4.4 Feature Extraction

An autonomous mobile robot must be able to determine its relationship to the environment by making measurements with its sensors and then using those measured signals. A wide variety of sensing technologies are available, as shown in section 4.1. But every sensor we have presented is imperfect: measurements always have error and, therefore, uncertainty associated with them. Therefore, sensor inputs must be used in a way that enables the robot to interact with its environment successfully in spite of measurement uncertainty.

There are two strategies for using uncertain sensor input to guide the robot's behavior. One strategy is to use each sensor measurement as a raw and individual value. Such raw sensor values could, for example, be tied directly to robot behavior, whereby the robot's actions are a function of its sensor inputs. Alternatively, the raw sensor values could be used to update an intermediate model, with the robot's actions being triggered as a function of this model rather than of the individual sensor measurements.

The second strategy is to extract information from one or more sensor readings first, generating a higher-level *percept* that can then be used to inform the robot's model and perhaps the robot's actions directly. We call this process *feature extraction*, and it is this next, optional step in the perceptual interpretation pipeline (figure 4.63) that we will now discuss.

In practical terms, mobile robots do not necessarily use feature extraction and scene interpretation for every activity. Instead, robots will interpret sensors to varying degrees depending on each specific functionality. For example, in order to guarantee emergency stops in the face of immediate obstacles, the robot may make direct use of raw forward-facing range readings to stop its drive motors. For local obstacle avoidance, raw ranging sensor strikes may be combined in an occupancy grid model, enabling smooth avoidance of obstacles meters away. For map-building and precise navigation, the range sensor values and even vision-sensor measurements may pass through the complete perceptual pipeline,

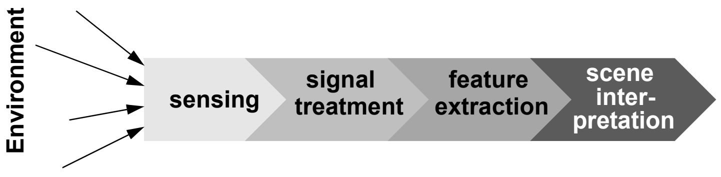


Figure 4.63
The perceptual pipeline: from sensor readings to knowledge models.

being subjected to feature extraction followed by scene interpretation to minimize the impact of individual sensor uncertainty on the robustness of the robot's mapmaking and navigation skills. The pattern that thus emerges is that, as one moves into more sophisticated, long-term perceptual tasks, the feature-extraction and scene-interpretation aspects of the perceptual pipeline become essential.

Feature definition. Features are recognizable structures of elements in the environment. They can usually be extracted from measurements and mathematically described. Good features are always perceivable and easily detectable from the environment. We distinguish between *low-level features* (*geometric primitives*) such as lines, points, corners, blobs, circles, or polygons and *high-level features* (*objects*) such as doors, tables, or trash cans. At one extreme, raw sensor data provide a large volume of data, but with low distinctiveness of each individual quantum of data. Making use of raw data has the potential advantage that every bit of information is fully used, and thus there is a high conservation of information. Low-level features are abstractions of raw data, and as such they provide a lower volume of data while increasing the distinctiveness of each feature. The hope, when one incorporates low-level features, is that the features are filtering out poor or useless data, but of course it is also likely that some valid information will be lost as a result of the feature-extraction process. High-level features provide maximum abstraction from the raw data, thereby reducing the volume of data as much as possible while providing highly distinctive resulting features. Once again, the abstraction process has the risk of filtering away important information, potentially lowering data utilization.

Although features must have some spatial locality, their geometric extent can range widely. For example, a corner feature inhabits a specific coordinate location in the geometric world. In contrast, a visual "fingerprint" identifying a specific room in an office building applies to the entire room, but it has a location that is spatially limited to the one particular room.

In mobile robotics, features play an especially important role in the creation of environmental models. They enable more compact and robust descriptions of the environment, helping a mobile robot during both map-building and localization. When designing a mobile robot, a critical decision revolves around choosing the appropriate features for the robot to use. A number of factors are essential to this decision.

Target environment. For geometric features to be useful, the target geometries must be readily detected in the actual environment. For example, line features are extremely useful in office building environments due to the abundance of straight wall segments, while the same features are virtually useless when navigating on Mars. Conversely, point features (such as corners and blobs) are more likely to be found in any textured environment. As an example, consider that the two NASA Mars explorations rovers, Spirit and Opportunity, used corner features (section 4.5) for visual odometry [203].

Available sensors. Obviously, the specific sensors and sensor uncertainty of the robot impacts the appropriateness of various features. Armed with a laser rangefinder, a robot is well qualified to use geometrically detailed features such as corner features owing to the high-quality angular and depth resolution of the laser scanner. In contrast, a sonar-equipped robot may not have the appropriate tools for corner feature extraction.

Computational power. Visual feature extraction can effect a significant computational cost, particularly in robots where the vision sensor processing is performed by one of the robot's main processors.

Environment representation. Feature extraction is an important step toward scene interpretation, and by this token the features extracted must provide information that is consonant with the representation used for the environmental model. For example, nongeometric visual features are of little value in purely geometric environmental models but can be of great value in topological models of the environment. Figure 4.64 shows the application of two different representations to the task of modeling an office building hallway. Each approach has advantages and disadvantages, but extraction of line and corner features has much more relevance to the representation on the left. Refer to chapter 5, section 5.5 for a close look at map representations and their relative trade-offs.

In sections 4.5–4.7, we present specific feature extraction techniques based on the two most popular sensing modalities of mobile robotics: vision and range sensing.

Visual interpretation is, as we have mentioned before, an extremely challenging problem to fully solve. Significant research effort has been dedicated over the past several decades to inventing algorithms for understanding a scene based on 2D images, and the research efforts have slowly produced fruitful results.

In section 4.2 we saw vision ranging and color-tracking sensors that are commercially available for mobile robots. These specific vision applications have witnessed commercial

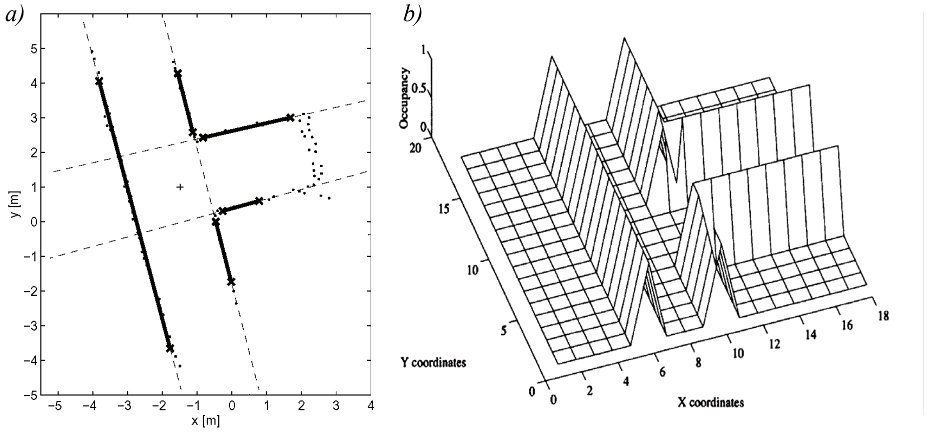


Figure 4.64
Environment representation and modeling: (a) feature-based (continuous metric); (b) occupancy grid (discrete metric). Courtesy of Sjur Vestli.

solutions primarily because the challenges are in both cases relatively well focused and the resulting, problem-specific algorithms are straightforward. But images contain much more than implicit depth information and color blobs. We would like to solve the more general problem of extracting a large number of feature types from images.

The next section presents some point feature extraction techniques that are relevant to mobile robotics along these lines. Two key requirements must be met for a visual feature extraction technique to have mobile robotic relevance. First, the method must operate in real time. Mobile robots move through their environment, and so the processing simply cannot be an offline operation. Second, the method must be robust to the real-world conditions outside of a laboratory. This means that carefully controlled illumination assumptions and carefully painted objects are unacceptable requirements.

Throughout the following descriptions, keep in mind that vision interpretation is primarily about the challenge of *reducing information*. A sonar unit produces perhaps fifty bits of information per second. By contrast, a CCD camera can output 240 *million* bits per second! The sonar produces a tiny amount of information from which we hope to draw broader conclusions. But the CCD chip produces too much information, and this overabundance of information mixes together relevant and irrelevant information haphazardly. For example, we may intend to measure the color of a landmark. The CCD camera does not simply report its color, but also measures the general illumination of the environment, the direction of illumination, the defocusing caused by optics, the side effects imposed by nearby objects with different colors, and so on. Therefore, the problem of visual feature extraction is largely one of removing the majority of irrelevant information in an image so that the remaining information unambiguously describes specific features in the environment.

4.5 Image Feature Extraction: Interest Point Detectors

In this section, we define the concept of the local feature and review some of the most consolidated feature extractors. As the computer vision literature in this field is very large, we will only describe in detail the two most popular feature detectors, namely Harris and SIFT, and will briefly introduce the others by explaining the main advantages and disadvantages and domain of application. For the interested reader, a comprehensive survey on local feature detectors can be found in [320].

4.5.1 Introduction

A local feature is an image pattern that differs from its immediate neighborhood in terms of intensity, color, and texture. Local features can be small image patches (such as regions of uniform color), edges, or points (such as corners originated from line intersections). In the modern terminology, local features are also called interest points, interest regions, or keypoints.

Depending on their semantic content, local features can be divided into three different categories. In the first category are features that have a semantic interpretation such as, for instance, edges corresponding to lanes of the road or blobs corresponding to blood cells in medical images. This is the case in most automotive applications, airborne images, and medical image processing. Furthermore, these were also the first applications for which local feature detectors have been proposed. In the second category are features that do not have a semantic interpretation. Here, what the features actually represent is not relevant. What matters is that their location can be determined accurately and robustly over time. Typical applications are feature tracking, camera calibration, 3D reconstruction, image mosaicing, and panorama stitching. Finally, in the third category are features that still do not have a semantic interpretation if taken individually, but that can be used to recognize a scene or an object if taken all together. For instance, a scene could be recognized counting the number of feature matches between the observed scene and the query image. In this case, the location of the feature is not important; only the number of matches is relevant. Application domains include texture analysis, scene classification, video mining, and image retrieval (see, for instance, Google Images, Microsoft Bing Images, Youtube, or Tineye.com). This principle is the basis of the *visual-word*-based place recognition that will be described in section 4.6.

4.5.2 Properties of the ideal feature detector

In this section we summarize the properties that an ideal feature detector should have. Let us start with a concrete example from digital image photography. Most of today's digital consumer cameras come with software for automatic stitching of panoramas from multiple photos. An example is shown in figure 4.65. The user simply takes several shots of the scene with little overlap between adjacent pictures and the software automatically aligns and fuses them all together into a cylindrical panorama (figure 4.65a). The key challenge is to identify corresponding regions between overlapping images. As the reader may perceive, one way to solve this problem is to extract feature points from adjacent pictures, find corresponding pairs according to some similarity measure (figure 4.65b), and compute the transformation (e.g., homography) to align them (figure 4.65c). The first problem is how to detect the same points independently in both images. In figure 4.65d, for instance, the features from the left image are not redetected in the right image. Because of this, we need a "repeatable" feature detector. The second problem is: for each point in the first image we need to correctly recognize the corresponding one in the second image. Thus, the detected features should be very distinctive (i.e., highly distinguishable).

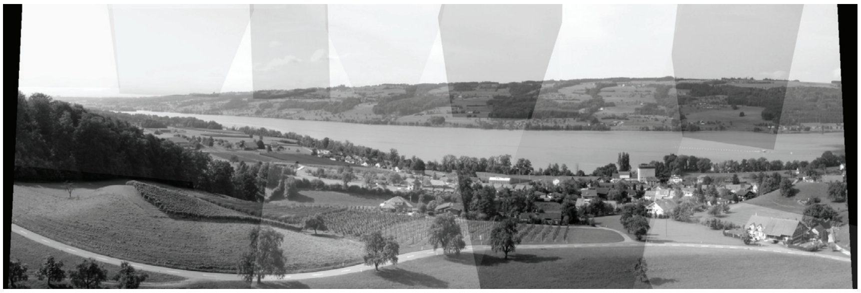
"Repeatability" is probably the most important property of a good feature detector. Given two images of the same scene taken under different viewing and illumination conditions, it is desirable that a high percentage of the features of the first image can be redetected in the second image. This requires the feature be invariant to view point changes, such as camera rotation or zoom (i.e., scale), and illumination changes.

The second important property is "distinctiveness," that is, the information carried by the patch surrounding the feature point should be as distinctive as possible so that the features can be distinguished and matched. For instance, the corners of a chessboard are not distinctive because they cannot be distinguished from each other. As we will see later, distinctiveness is also the main difference between Harris and SIFT features. Harris privileges corners (such as edge intersections), while SIFT privileges image patches with highly informative content (i.e., not corners).

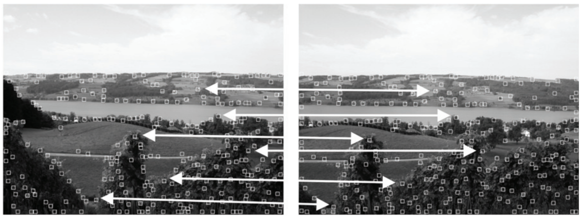
Other important properties of a good feature detector are:

- Localization accuracy: the detected features should be accurately localized, both in image position and scale. Accuracy is especially important in camera calibration, 3D reconstruction from images ("structure from motion"), and panorama stitching.

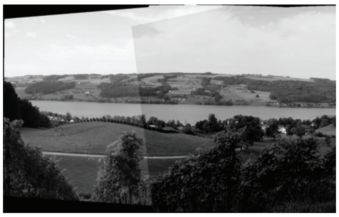
- Quantity of features: the ideal number of detected features depends on the application. For most of the tasks like object or scene recognition, image retrieval, and 3D reconstruction it is important to have a sufficiently large number of features, this in order to increase the recognition rate or the accuracy of the reconstruction. However, if the feature had a semantic interpretation, then a small number of features would be enough to



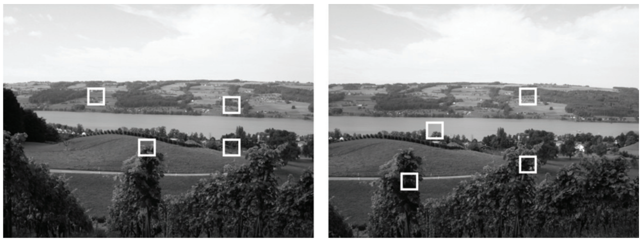
(a)



(b)



(c)



(d) No chance to match!

Figure 4.65 (a) Panorama built from multiple overlapping images using Autostitch software. (b) First step: select salient features in both images and match corresponding ones. (c) Second step: compute the transformation between the two corresponding sets and align the images. (d) Two example images where features were not redetected and therefore there is no chance to match them.

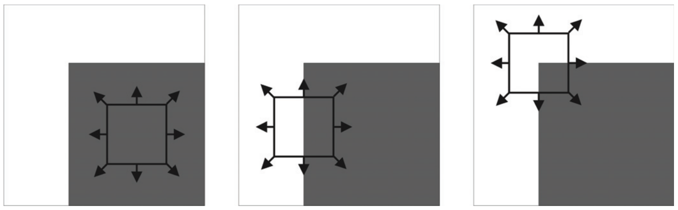


Figure 4.66 (a) “Flat” region: no change in all directions. (b) “Edge”: no change along the edge direction. (c) “Corner”: significant change in all directions.

recognize a scene (as an example, some semantic “high level” features could be individual objects or objects parts such as table, chair, table leg, door, and so on).

- Invariance: good features should be invariant to changes of camera viewpoint, environment illumination, and scale (like zoom or camera translation). Invariance can be achieved within a certain range when these changes can be modeled as mathematical transformations (see section 4.5.4). A successful result in this direction has been successfully demonstrated by some of the recent detectors like SIFT (section 4.5.5.1).
- Computational efficiency: it is also desirable that features can be detected and matched very efficiently. In the framework of the project undertaken by Google Images of organizing all the images on the web, computation efficiency is a critical component as its database—nowadays composed of billions of images—grows more and more every year. This is even important in robotics, where most of the applications need to work in real-time. However, the time of detection and matching of a feature is strictly related to the degree of invariance desired: the higher the level of invariance, the more image transformations to check, and, thus, the longer the computation time.
- Robustness: the detected features should be robust to image noise, discretization effects, compression artifacts, blur, deviations from the mathematical model used to obtain invariance, and so on.

4.5.3 Corner detectors

A corner in an image can be defined as the intersection of two or more edges. Corners are features with high repeatability.

The basic concept of corner detection. One of the earliest corner detectors was invented by Moravec [234, 235]. He defined a corner as a point where there is a large intensity variation in every direction. An intuitive explanation of his corner detection algorithm is given in figure 4.66. Intuitively, one could recognize a corner by looking through a small window centered on the pixel. If the pixel lies in a “flat” region (i.e., a region of uniform intensity), then the adjacent windows will look similar. If the pixel is along an edge, then adjacent windows in the direction perpendicular to the edge will look different, but adjacent windows in a direction parallel to the edge will result only in a small change. Finally, if the pixel lies on a corner, then none of the adjacent windows will look similar. Moravec used the Sum of Squared Differences (SSD, section 4.3.3) as a measure of the similarity between two patches. A low SSD indicates more similarity. If this number is locally maximal, then a corner is present.

4.5.3.1 The Harris corner detector

Harris and Stephens [146] improved Moravec's corner detector by considering the partial derivatives of the SSD instead of using shifted windows.

Let I be a grayscale image. Consider taking an image patch centered on (u, v) and shifting it by (x, y). The Sum of Squared Differences SSD between these two patches is given by:

$$SSD(x, y) = \sum_u \sum_v ((I(u, v)) - I(u + x, v + y))^2 . \tag{4.116}$$

$I(u + x, v + y)$ can be approximated by a first-order Taylor expansion. Let I_x and I_y be the partial derivatives of I, such that

$$I(u + x, v + y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y . \tag{4.117}$$

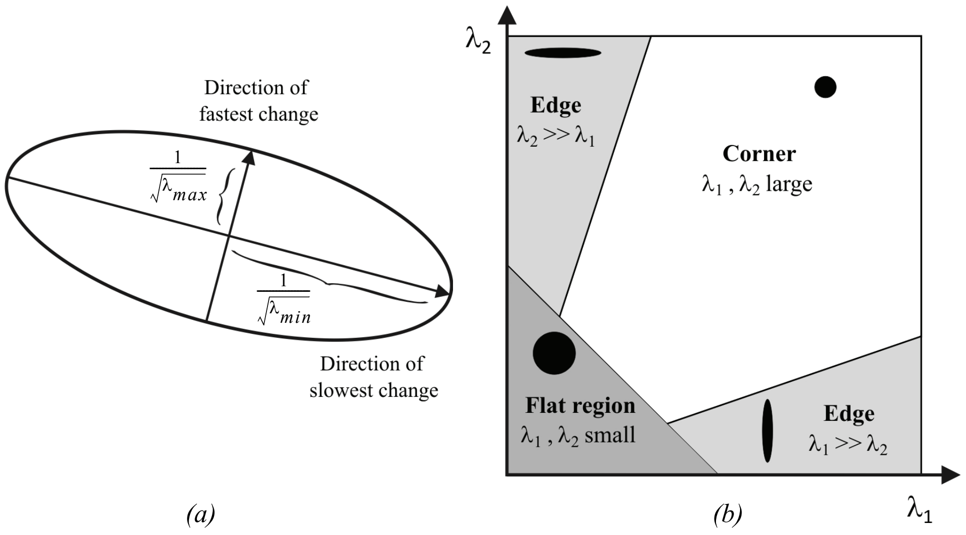
This produces the approximation

$$SSD(x, y) \approx \sum_u \sum_v (I_x(u, v)x + I_y(u, v)y)^2 , \tag{4.118}$$

which can be written in matrix form:

$$SSD(x, y) \approx \begin{bmatrix} x & y \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix} , \tag{4.119}$$

where M is the second moment matrix



Direction of fastest change
$\frac{1}{\sqrt{\lambda_{max}}}$
$\frac{1}{\sqrt{\lambda_{min}}}$
Direction of slowest change
(a)
λ_2
Edge
$\lambda_2 >> \lambda_1$
Corner
λ_1, λ_2 large
Flat region
λ_1, λ_2 small
Edge
$\lambda_1 >> \lambda_2$
(b)
λ_1

Figure 4.67 (a) This ellipse is built from the second moment matrix and visualizes the directions of fastest and lowest intensity change. (b) The classification of corner and edges according to Harris and Stephens.

$$M = \sum_u \sum_v \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum\sum I_x^2 & \sum\sum I_x I_y \\ \sum\sum I_x I_y & \sum\sum I_y^2 \end{bmatrix}. \tag{4.120}$$

And since M is symmetric, we can rewrite M as

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R, \tag{4.121}$$

where λ_1 and λ_2 are the eigenvalues of M.

As mentioned before, a corner is characterized by a large variation of SSD in all directions of the vector (x, y). The Harris detector analyses the eigenvalues of M to decide if we are in presence of a corner or not. Let us first give an intuitive explanation before showing the mathematical expression.

Using equation (4.119) we can visualize M as an ellipse (figure 4.67a) of equation:

$$\begin{bmatrix} x & y \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix} = const. \tag{4.122}$$

The axis lengths of this ellipse are determined by the eigenvalues of M and the orientation is determined by R.

Based on the magnitudes of the eigenvalues, the following inferences can be made based on this argument:

- If both λ_1 and λ_2 are small, SSD is almost constant in all directions (i.e., we are in presence of a flat region).
- If either $\lambda_1 \gg \lambda_2$ or $\lambda_2 \gg \lambda_1$, we are in presence of an edge: SSD has a large variation only in one direction, which is the one perpendicular to the edge.
- If both λ_1 and λ_2 are large, SSD has large variations in all directions and then we are in presence of a corner.

The three situations mentioned above are pictorially summarized in figure 4.67b.

Because the calculation of the eigenvalues is computationally expensive, Harris and Stephens suggested the use of the following "cornerness function" instead:

$$C = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = det(M) - \kappa \cdot trace^2(M), \tag{4.123}$$

where κ is a tunable sensitivity parameter. This way, instead of computing the eigenvalues of M, we just need to evaluate the determinant and trace of M. The value of κ has to be determined empirically. In the literature, values are often reported in the range 0.04–0.15.

The last step of the Harris corner detector consists in extracting the local maxima of the cornerness function, using *nonmaxima suppression*[14]. Finally only the local maxima which are above a given threshold are retained. The processing steps are illustrated in figure 4.68.

Figure 4.68c shows the corners detected for the two example images. Notice that the images are related by a rotation and a slight change of illumination. As can be seen, many of the features detected in the left image have also been redetected in the right image. This means that the repeatability of the Harris detector under rotations and small changes of illumination is high. In the next section, we will point out the properties of the Harris detector as well as its drawbacks.

14. *Nonmaxima suppression* involves revisiting every pixel of the cornerness function and determining whether or not it is at a local maximum. If not, then the value is set to zero. This causes only the maxima to be preserved.

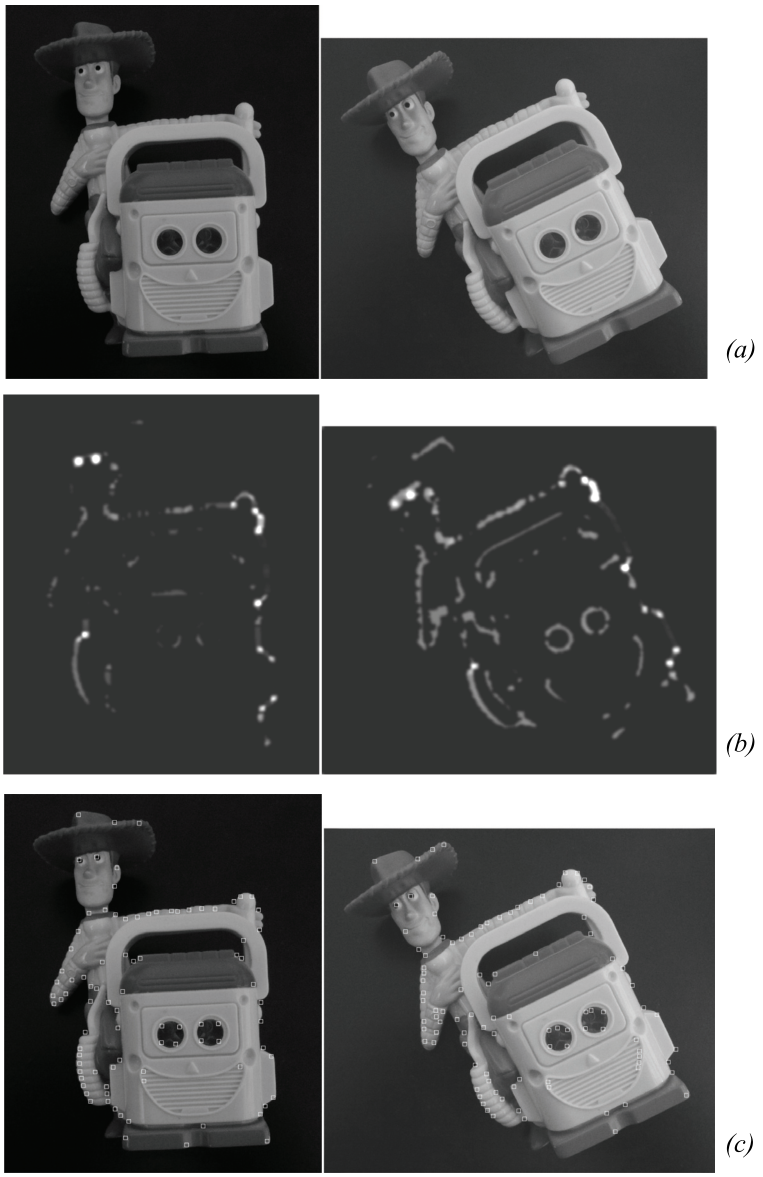


Figure 4.68 Extraction of the Harris corners. (a) Original image. (b) Cornerness function. (c) Harris corners are identified as local maxima of the cornerness function (only local maxima larger than a given threshold are retained). Two images of the same object are shown, which differ by illumination and orientation.

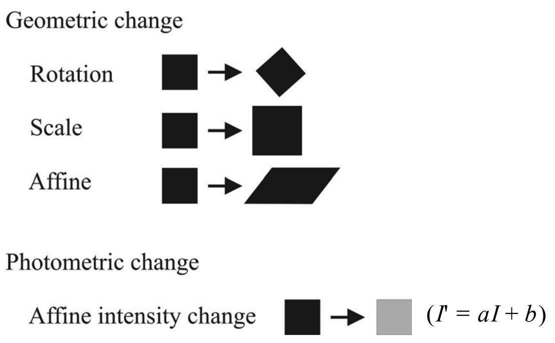


Figure 4.69 Models of image changes.

4.5.4 Invariance to photometric and geometric changes

As observed in section 4.5.2, in general we want features to be detected despite geometric and photometric changes in the image: if we have two transformed versions of the same image, features should be detected in corresponding locations. Image transformations can affect the geometric or the photometric properties of an image. Consolidated models of image transformations are the following (see also figure 4.69):

- Geometric changes:
  - 2D Rotation
  - Scale (uniform rescaling)
  - Affine
- Photometric changes
  - Affine intensity

Observe that we did not mention changes of camera viewpoint (i.e., perspective distortions). In this case the transformation valid only for planar objects would be a homography. However, when the viewpoint changes are small and the object is locally planar, the affine transformation is a good approximation of the homography. Observe also that 2D rotation occurs only if the camera rotates purely about its optical axis. Uniform rescaling, instead, appears when the camera zooms (in or out) or translates along the direction of its optical axis, but the latter is valid only for locally planar objects.

As an example, let us now examine the invariance of the Harris detector to the above mentioned transformations. We can observe that the Harris detector is invariant to 2D image rotations. This can be explained by observing that the eigenvalues of the second moment matrix do not change under pure rotation. Indeed, the ellipse rotates, but its shape

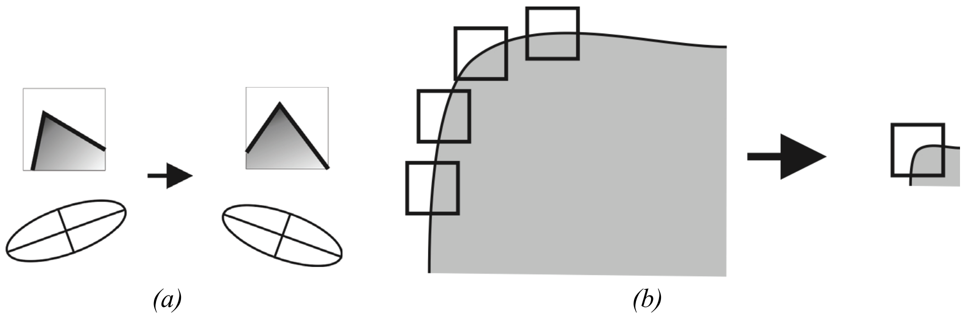


Figure 4.70 (a) Harris detector is invariant to image rotations: the ellipse rotates but its shape (i.e. eigenvalues) remains the same. (b) Conversely, it is not invariant to image scale: at a large scale (left) all points along the corner would be classified as edges, while at a smaller scale case (right) the point would be classified as corner.

(i.e., the eigenvalues) remains the same (see figure 4.70a). As a matter of fact, observe that to make the Harris detector isotropic (i.e., uniform for all rotations), the second moment matrix should be computed in a circular region rather in a squared window. This is usually done by averaging equation (4.120) with a circular symmetric Gaussian function.

The Harris detector is also invariant to affine intensity changes. In this case, the eigenvalues, and so the cornerness function, are rescaled by a constant factor, but the position of the local maxima of the cornerness function remains the same. Conversely, the Harris detector is not invariant to geometric affine transformations or scale changes. Intuitively, an affine transformation distorts the neighborhood of the feature along the x and y directions and, accordingly, a corner can get reduced or increased its curvature. Regarding scale changes, this is immediately clarified as observed in figure 4.70b. In this figure, the corner would be classified as an edge at a high scale and as a corner at a smaller scale.

The performance of the Harris detector against scale changes is shown in figure 4.71 according to a comparative study made in [216]. In this figure, the repeatability rate is plotted versus the scale factor. The repeatability rate between two images is computed as the ratio between the number of found correspondences and the number of all possible correspondences. As observed, after rescaling the image by a factor 2, only 20% of the possible correspondences are redetected.

Although it is not invariant to scale changes, the Harris detector, as in its original implementation by Harris and Stephens, is still widely used and can be found in the well-known Intel open-source computer vision library (OpenCV [343]). Furthermore, in a comparative study of different interest point detectors, Schmid et al. [285] showed that the Harris corners are among the most repeatable and most informative features. As we will see in the next sections, some modifications to the original implementation have made the Harris detector also invariant to scale and affine changes. Additionally, the location accuracy of

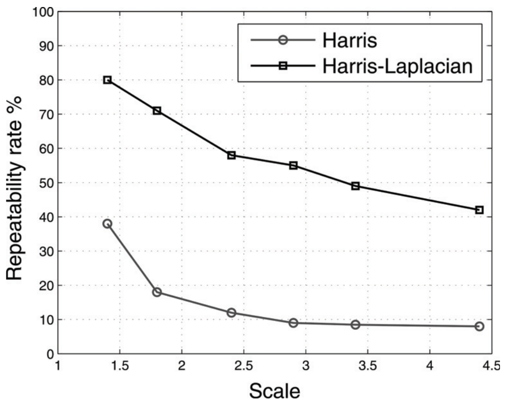


Figure 4.71 Repeatability rate of Harris detector. Comparison with Harris-Laplacian. This plot is the result of a comparative study presented in [216].

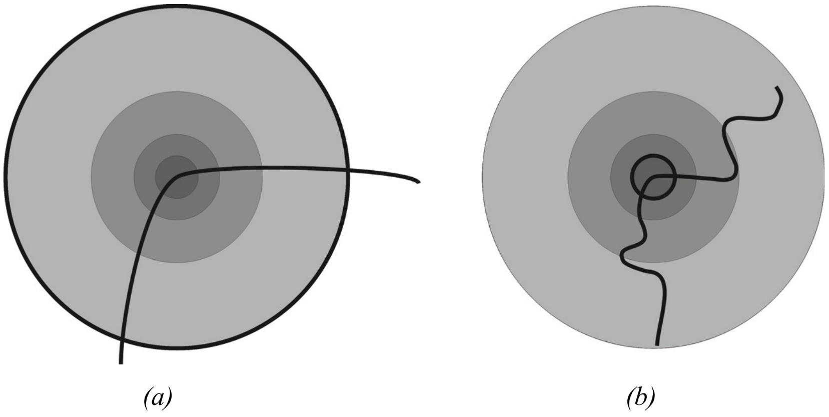


Figure 4.72 To achieve scale-invariant detection, the image is analyzed at different scales. This means, the Harris detector is applied several times on the image, each time with a different window size.

the Harris corners can be improved up to subpixel precision. This can be achieved by approximating the cornerness function in the neighborhood of a local maximum through a quadratic function.

4.5.4.1 Scale-invariant detection

In this section, we will describe the modifications that have been devised to make the Harris detector invariant to scale changes. If we look at figure 4.72, we will notice that one way to detect the corner at higher scale is to use a multiscale Harris detector. This means that

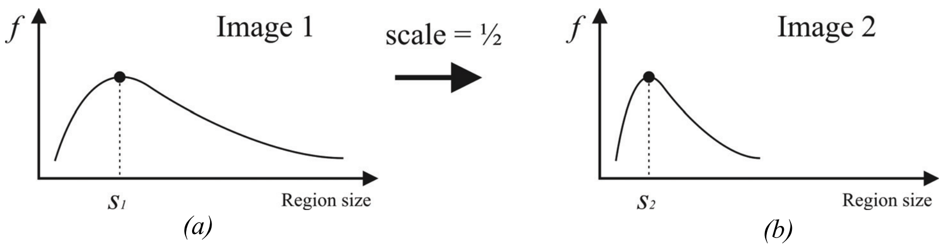


Figure 4.73 Average intensity as a function of the region size. (a) Original image. (b) Resized image.

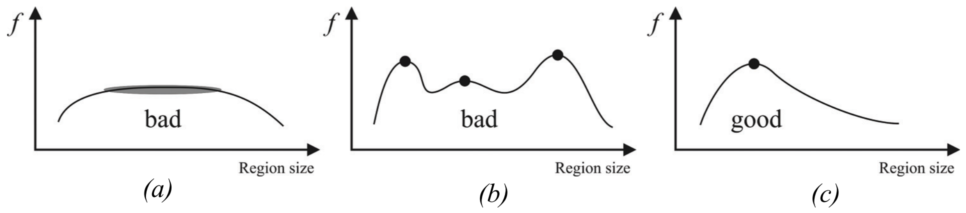


Figure 4.74 (a)-(b) are bad scale invariant functions. (c) is a good.

the same detector is applied several times on the image, each time with a different window (circle) size. Note, an efficient implementation of multiscale detection uses the so called scale-space pyramid: instead of varying the window size of the feature detector, the idea is to generate upsampled or downsampled versions of the original image (i.e., pyramid). Using multiscale Harris detector, we can be sure that at some point the corner of image 4.72a will get detected. Once the point has been detected in both images a question arises as: how do we select the corresponding scale? In other words, how do we choose corresponding circles independently in each image?

In computer vision, the correct-scale selection is usually done by following the approach proposed in 1998 by Lindeberg [193]: the appropriate scale of a local feature can be chosen as the one for which a given function reaches a maximum or minimum over scales. Let us give an intuitive explanation. For every circle in the two images in figure 4.72, let us plot the average intensity of the pixels within the circle as a function of the circle size. For these images, we will get the two plots shown in figure 4.73. As expected, these two functions look the same up to a rescaling in the x-axis. The solution to our problem is therefore to take as corresponding scales the circle sizes for which these functions reach their maximum. Depending of the chosen function, we might take the minimum instead of the maximum.

The problem is how to design a good function. The first observation we can make is that a "good" function for scale detection should have one stable sharp peak like that in figure 4.74. Despite the one used in our example, the average intensity is not good because it can

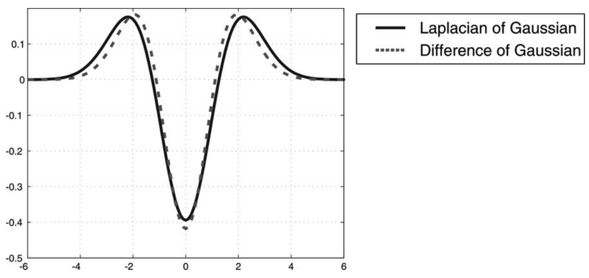


Figure 4.75 Comparison between Laplacian of Gaussian and Difference of Gaussian

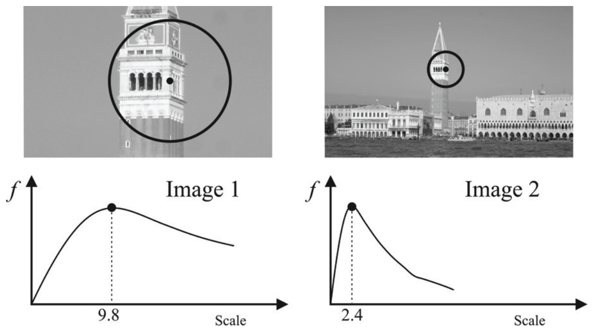


Figure 4.76 Response of the LOG operator over two corresponding points from two images taken at different scales.

return multiple peaks or even no peaks at all. As a matter of fact, it turns out that for usual images a good function is one that corresponds to contrast, that is, sharp local intensity changes. The Laplacian of Gaussian (LoG) (figure 4.75) is a good operator for identifying sharp intensity changes and is currently the one used for scale selection by the Harris corner detector. In a comparative study presented in [216, 101], the LoG operator has been shown to give the best results with respect to other functions. The response of the LoG over two image features is shown in figure 4.76.

The multiscale Harris detector is known as Harris-Laplacian and was implemented by Mikolajczyk and Schmid [217]. The comparison between the standard Harris and the Harris-Laplacian over scale is shown in figure 4.71.

4.5.4.2 Affine invariant detection

As mentioned in section 4.5.4, affine transformation is a good approximation of perspective distortion of locally planar patches under small viewpoint changes. In the previous section, we considered the problem of detection under uniform rescaling. The problem now is how

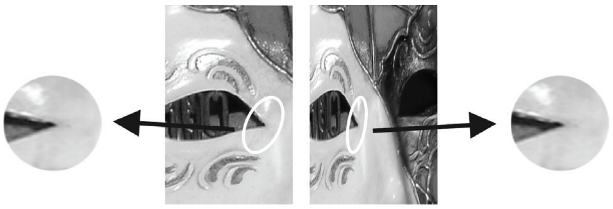


Figure 4.77 Computation of the affine invariant ellipse in two images related by an affine transformation.

to detect the same features under affine transformation, which can be seen as a nonuniform rescaling. The procedure consists in the following steps:

- First, the features are identified using the scale invariant Harris-Laplacian detector.
- Then, the second moment matrix (4.120) is used to identify the two directions of slowest and fastest change of intensity around the feature.
- Out of these two directions, an ellipse is computed to the same size as the scale computed with the LoG operator.
- The region inside the ellipse is normalized to a circular one.
- The initial detected ellipse and the resulting normalized circular shape are shown in figure 4.77.

The affine invariant Harris detector is known as Harris-Affine and was devised by Mikolajczyk and Schmid [218].

4.5.4.3 Other corner detectors

The Shi-Tomasi corner detector. This is also sometimes referred to as the Kanade-Tomasi corner detector [284]. This detector is strongly based on the Harris corner detector. The authors show that for image patches undergoing affine transformations, $min(\lambda_1, \lambda_2)$ is a better measure than the cornerness function C (4.123).

The SUSAN corner detector. SUSAN stands for Smallest Univalue Segment Assimilating Nucleus and, besides being used for corner detection, it is also used for edge detection and noise suppression. The SUSAN corner detector has been introduced by Smith and Brady [296]. Its working principle is different from the Harris detector. As we have seen, Harris is based on local image gradients, which are computationally expensive to compute. Conversely, SUSAN is based on a morphological approach, which is computationally much more efficient than Harris.

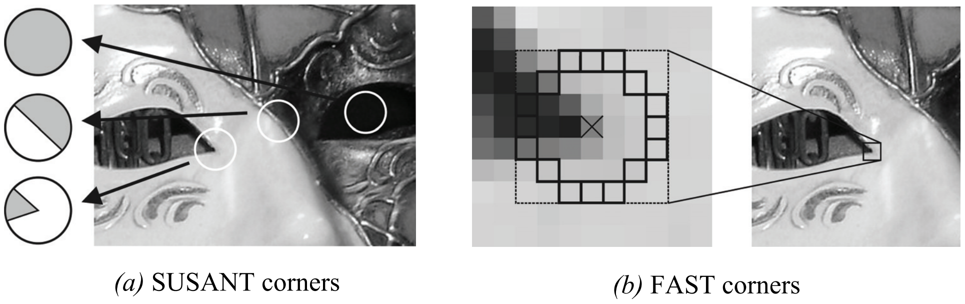


Figure 4.78 (a) SUSAN detector compares pixels within a circular region, while FAST (b) compares them only on a circle.

The working principle of SUSAN is very simple (see also figure 4.78a). For each pixel in the image, SUSAN considers a circular window of fixed radius centered on it. Then, all the pixels within this window are divided into two categories, depending on whether they have "similar" or "different" intensity values as the center pixel. Accordingly, on uniform intensity regions of the image, most pixels within the window will have a similar brightness as the center pixel. Near edges, the fraction of pixels with similar intensity will drop to 50%, while near corners it will decrease further to about 25%. Thus, SUSAN corners are identified as image locations where the number of pixels with similar brightness in a local neighborhood attains a local minimum and is below a specified threshold. As a final step, nonmaxima suppression (page 218) is used to identify local minima.

The SUSAN corners show a high repeatability, however they are heavily sensitive to noise. Indeed, many of the features are often located on edges than on real corners.

The FAST corner detector. The FAST (Features from Accelerated Segment Test) detector, was introduced by Rosten and Drummond [267, 268]. This detector builds upon the SUSAN detector. As we have seen, SUSAN computes the fraction of pixels within a circular window, which have similar intensity as the center pixel. Conversely, FAST compares pixels only on a circle of 16 pixels around the candidate corner (see figure 4.78b). This results in a very efficient detector that is up to thirty times faster than Harris: FAST takes only 1–2 milliseconds on a 2GHz Dual Core laptop and is currently the most computationally efficient feature detector available. However, like the SUSAN, it is not robust at high levels of noise.

4.5.4.4 Discussion about corner detectors

The Harris detector, with its scale and affine invariant extensions, is a convenient tool for extracting a large number of corners. Additionally, it has been identified as the most stable corner detector, as reported in several evaluations [28, 219, 285]. Alternatively, the

SUSAN or the FAST detectors can be used. They are much more efficient but also more sensitive to noise.

Shi-Tomasi, SUSAN, and FAST can also be made scale invariant like the Harris-Laplacian by analyzing the image at multiple scales, as seen in section 4.5.4.1. However, the scale estimation of corners is less accurate than blobs (e.g., SIFT, MSER, or SURF) due to the multiscale nature of corners: by definition, a corner is found at the intersection of edges, therefore its appearance changes very little at adjacent scales.

Finally, it is important to remind that the affine transformation model holds only for small viewpoint changes and in case of locally planar regions, that is, assuming the camera is relatively far from the object.

In the next section, we will describe the SIFT detector. Despite being a blob detector, the SIFT features incorporate all the properties of the scale-affine-invariant Harris but they are much more distinctive and robust to image noise, small illumination changes, and large changes of camera viewpoint.

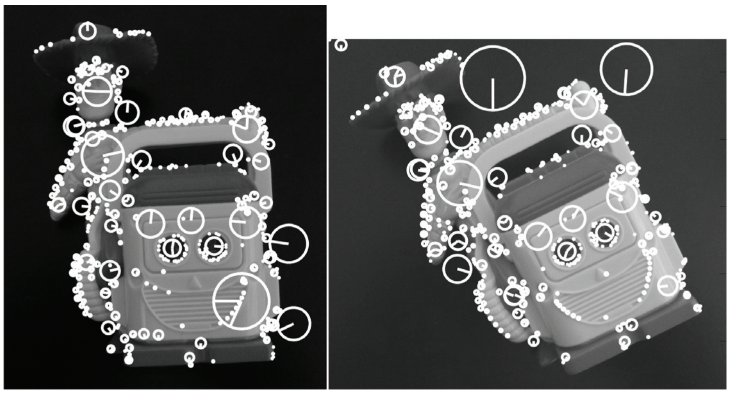
4.5.5 Blob detectors

A blob is an image pattern that differs from its immediate neighborhood in terms of intensity, color, and texture. It is not an edge, nor a corner. The location accuracy of a blob is typically smaller than that of a corner, but its scale and shape are better defined. To be clearer, a corner can be localized by a single point (e.g., the intersection of two edges), while a blob can only be localized by its boundary. On the other hand, a corner is less accurately localized over the scale because, as we pointed out before, a corner is found at the intersection of edges and therefore its appearance changes very little at adjacent scales. Conversely, a blob is more accurately localized over the scale because the boundary of a blob defines immediately its size and so its scale.

Using the new terminology, blob detectors can also be referred to as interest point operators, or alternatively interest region operators. Some examples of bloblike features are shown in figure 4.79. In this figure you can see two feature types that will be described in this section, namely SIFT and MSER. As observed, MSER privileges regions with uniform intensity, while SIFT does not.

4.5.5.1 SIFT features

SIFT stands for Scale Invariant Feature Transform and is a method to detect and match robust keypoints, which was invented in 1999 by Lowe [196, 197]. The uniqueness of SIFT is that these features are extremely distinctive and can be successfully matched between images with very different illumination, rotation, viewpoint, and scale changes. Its high repeatability and high matching rate in very challenging conditions have made SIFT the best feature detector so far. It has found many applications in object recognition, robotic


(a) SIFT features

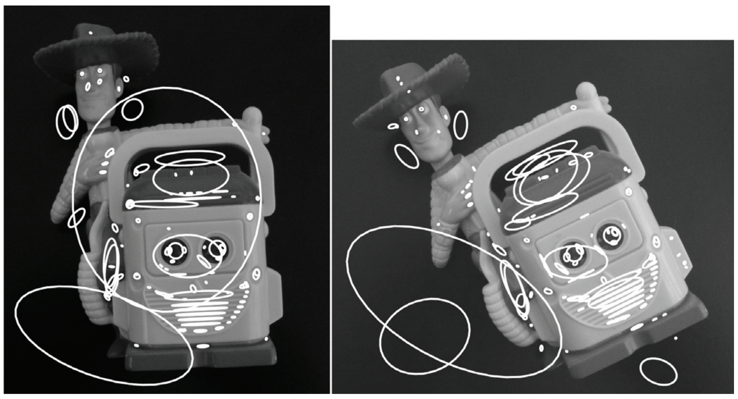

(b) MSER features

Figure 4.79 Extraction of SIFT and MSER features from the same sample image used for the Harris detector in figure 4.68. Observe that both SIFT and MSER avoid corners. Futhemore, MSER privileges regions with uniform intensity. Both these feature detectors are robust to large changes of intensity, scale, and viewpoint.

mapping and navigation, image stitching (e.g. panoramas, mosaics), 3D modeling, gesture recognition, video tracking, and face recognition.

The main advantage of the SIFT features in comparison to all previously explained methods is that a "descriptor" is computed from the region around the interest point, which distinctively describes the information carried by the feature. As we will see, this descriptor is a vector that represents the local distribution of the image gradients around the interest point. As proven by its inventor, it is actually this descriptor that makes SIFT robust to rotation and small changes of illumination, scale, and viewpoint.

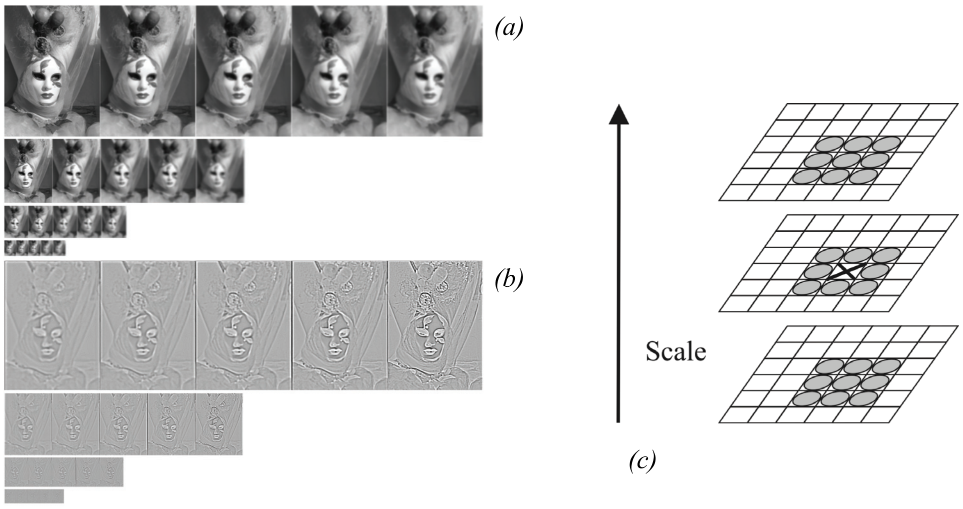


Figure 4.80 (a) Gaussian blurred images at different scales. (b) Difference of Gaussian images. (c) Keypoint selection as local maxima or minima of the DoG images across adjacent scales.

In the following, we will analyze the main steps of the SIFT algorithm, which are:

- Identification of keypoint location and scale
- Orientation assignment
- Generation of keypoint descriptor

Identification of keypoint location and scale. The first step toward the identification of SIFT keypoints is the generation of the so-called Difference of Gaussian (DoG) images. This is done by first blurring the original image with Gaussian filters at different scales (i.e., different sigma) and then by taking the difference of successive Gaussian-blurred images. This process is shown in figure 4.80a: the original image (top left) is blurred with four Gaussian filters with different sigma, and this is repeated after downsampling the image of a factor 2. Finally, DoG images are computed by simply taking the difference between successive blurred images 4.80b.

The second step is the selection of the keypoints. SIFT keypoints are identified as local maxima or minima of the DoG images across scales. In particular, each pixel in the DoG images is compared to its eight neighbors at the same scale, plus the nine neighbors at adjacent scales (figure 4.80c). If the pixel is a local maximum or minimum, it is selected as a candidate keypoint.

The third step consists in refining the location, in both space and scale, of the keypoints by interpolation of nearby data. Finally, keypoints with low contrast or along edges are removed because of their low distinctiveness and due to their instability to image noise.

Note that another way of generating DoG images consists in convolving the image with a DoG operator, which is nothing but the difference between to Gaussian filters (figure 4.75). As shown in figure 4.75, the DoG function is actually a very good approximation of the Laplacian of Gaussian (LoG). However, DoG images are more efficient to compute, and therefore they have been used in SIFT in lieu of LoG. At this point the attentive reader will recognize that the scale extrema selection of the SIFT is very similar to the scale extrema selection of the Harris-Laplacian. Indeed, the main difference with the Harris-Laplacian is the identification of the keypoint location. While in Harris the keypoint is identified in the image plane as local maximum of the cornerness function, in SIFT the keypoint is a local minimum or maximum of the DoG image in both position and scale. To recap, in SIFT the DoG operator is used to identify both position and scale of the keypoints.

Orientation assignment. This step consists in assigning each keypoint a specific orientation in order to make it invariant to image rotation.

To determine the keypoint orientation, a gradient orientation histogram is computed in the neighborhood of the keypoint. In other words, for every pixel in the neighborhood of the keypoints, the intensity gradient (magnitude and orientation) is computed. Then a histogram of orientations is built such that the contribution of each pixel is weighted by the gradient magnitude.

Peaks in the histogram correspond to dominant orientations (figure 4.81a). Once the histogram is filled, the orientation corresponding to the highest peak is assigned to the keypoint. In the case of multiple peaks that are within 80% of the highest peak, an additional keypoint is created for each additional orientation, having the same location and scale as the original keypoint. All the properties of the keypoint will be measured relative to the keypoint orientation. This provides invariance to rotation.

Final keypoints with selected orientation and scale are shown in figure 4.81b.

Generation of keypoint descriptor. In the previous steps, we have described how to detect SIFT keypoints in both location and scale spaces and how to assign orientations to them. The last step of the SIFT algorithm is to compute descriptor vectors for these keypoints such that the descriptors are highly distinctive and partially invariant to illumination and viewpoint.

The descriptor is based on gradient orientation histograms. In order to achieve orientation invariance, the gradient orientations are rotated relative to the keypoint orientation. The neighboring region of the keypoint is then divided into 4×4 smaller regions, and a gradient histogram with eight orientation bins is computed within each of these regions.

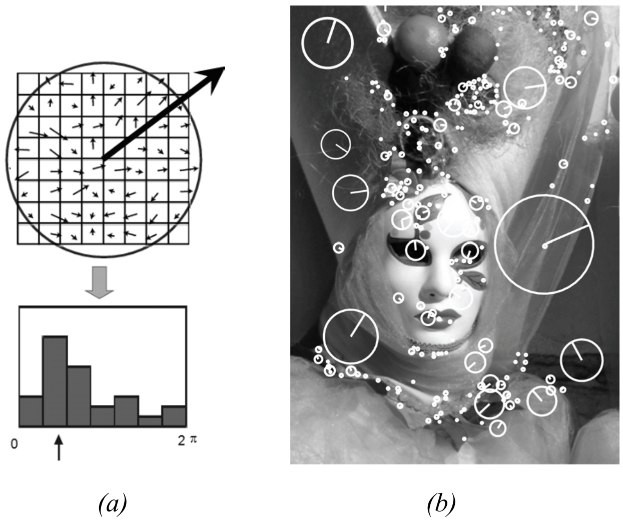


Figure 4.81 (a) Orientation assignment. (b) Some SIFT features with detected orientation and scale.

Finally, the descriptor is built by stacking all the orientation histogram entries. Therefore, the final length of the descriptor vector is $4 \times 4 \times 8 = 128$ elements. To achieve partial illumination invariance, the descriptor vector is finally normalized to have unit norm.

Observe, lower dimension descriptors could be built by using smaller region partitions or less histogram bins. However according to the literature, the 128 element vector is the one for which the best results in terms of robustness to image variations were reported.

In [197], it was also shown that feature matching accuracy is above 50% for viewpoint changes of up to 50 degrees (see figure 4.82). Therefore SIFT descriptors are invariant to minor viewpoint changes. To evaluate the distinctiveness of the SIFT descriptor, many tests were performed by counting the number of correct matches in a database with a varying number of keypoints. These tests revealed that matching accuracy of SIFT descriptor decreases only very little for very large database sizes. This implies that SIFT features are highly distinctive.

Because of its high repeatability and distinctiveness, SIFT has demonstrated in the last ten years to be the best feature detectors in a wide range of applications, although it is outperformed in efficiency by SURF (section 4.5.5.2). Excellent results have been achieved in robot navigation, 3D object recognition, place recognition, SLAM, panorama stitching, image retrieval, and many others.

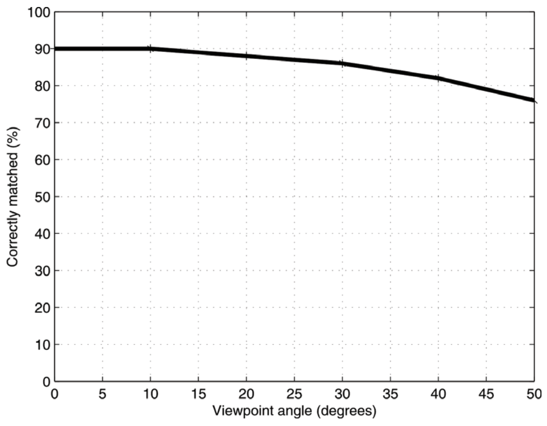


Figure 4.82 Fraction of SIFT keypoints correctly matched as a function of the viewpoint angle.

4.5.5.2 Other blob detectors

The MSER detector. Maximally Stable Extremal Regions (MSER) have been proposed by Matas et al. [210] for matching features that are robust under large viewpoint changes. A *maximally stable extremal region* is a connected component of pixels that have either higher or lower intensity than all the pixels on its outer boundary (figure 4.79b). These *extremal regions* are selected using an appropriate intensity thresholding and have a number of desirable properties. First, they are completely invariant to monotonic changes of intensity. Second, they are invariant to affine image transformations.

The SURF detector. SURF stands for Speeded Up Robust Features and have been proposed by Bay et al. [71]. This scale-invariant feature detector is strongly inspired by SIFT but is several times faster. Basically, it uses *Haar wavelets*[15] to approximate DoG filters and *integral images*[16] for convolution, which make the filtering process much more efficient at the expense of a minor robustness with respect to SIFT.

4.5.5.3 Summary on features detectors

Table 4.2 gives an overview of the most important properties for the feature detectors described in the previous sections. The highest repeatability and localization accuracy is

15. A Haar wavelet is a piecewise constant function.
16. Integral image is an algorithm for quickly and efficiently generating the sum of values in a rectangular subset of a grid.

Table 4.2
Comparison of feature detectors: properties and performance.

	Corner detector	Blob detector	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	x		x			+++	+++	++	++
Shi-Tomasi	x		x			+++	+++	++	++
Harris-Laplacian	x	x	x	x		+++	+++	++	+
Harris-Affine	x	x	x	x	x	+++	+++	++	++
SUSAN	x		x			++	++	++	+++
FAST	x		x			++	++	++	++++
SIFT		x	x	x	x	+++	++	+++	+
MSER		x	x	x	x	+++	+	+++	+++
SURF		x	x	x	x	++	++	++	++

obtained by the Harris detector and its scale and affine invariant versions. The SUSAN and FAST detectors avoid computation of image derivatives and are therefore more efficient than Harris but the absence of smoothing makes them more sensitive to noise. The original Harris, Shi-Tomasi, SUSAN, and FAST are not scale-invariant, however some literature exists on how achieving scale invariance using the approach described in section 4.5.4.1.

In contrast to the original Harris, Harris-Laplace attains scale invariance; however, its scale estimation is less accurate than SIFT, MSER, or SURF due to the multiscale nature of corners. Finally, the SURF detector shows high repeatability, scale, and viewpoint invariance. However, it was devised for efficiency, and therefore it does not perform as well as SIFT.

GPU and FPGA implementations. Some of these feature detectors have been implemented to take advantage of the parallelism offered by modern Graphics Processing Units (GPUs) and Field Programmable Gate Arrays (FPGA). GPU implementations of SIFT are described in [150, 292]. An FPGA implementation of the Harris-Affine feature detector is discussed in [89] and of the SIFT detector in [286]. The availability of these algorithms for GPUs and FPGA make computer vision algorithm able to work at high frame rates.

Open source software: Web resources.

- Most of the feature detectors described in this section (Harris, MSER, FAST, SURF, and several others) are available as ready-to-use code in the Intel open-source computer vision library (OpenCV): http://opencv.willowgarage.com/wiki
- SUSAN, original source code: http://users.fmrib.ox.ac.uk/~steve/susan
- FAST, original source code: http://mi.eng.cam.ac.uk/~er258/work/fast.html
- SIFT, the original executable from David Lowe: http://people.cs.ubc.ca/~lowe/keypoints
- A reimplementation of SIFT, MSER and other featured detectors by Andrea Vedaldi. Source code in C and Matlab: http://www.vlfeat.org
- 3D object recognition toolkit, based on SIFT. Developed at the Autonomous Systems Lab at the ETH Zurich: http://robotics.ethz.ch/~ortk
- ERSP Vision Tool. An exiting demo for live object recognition by Evolution Robotics: http://www.evolution.com/product/oem/download/?ch=Vision (it becomes downloadable after user registration)
- SURF, precompiled software (GPU implementation also available): http://www.vision.ee.ethz.ch/~surf

4.6 Place Recognition

4.6.1 Introduction

Location recognition (or *place recognition*) describes the capability of naming discrete places in the world. A requirement is that it is possible to obtain a discrete partitioning of the environment into places and a representation of the place and that the places with the corresponding representations are stored in a database. The location recognition process then works by computing a representation from the current sensor measurements of the robot and searching the database for the most similar representation stored. The retrieved representation then tells us the location of the robot.

Location recognition is the natural form of robot localization in a topological environment map as described by many authors [108, 131, 138, 208, 214, 325]. Visual sensors (i.e., cameras) are perfectly suited to create a rich representation that is both descriptive and discriminative. Most visual representations proposed so far can be divided into *global* representations and *local* representations. Global representations use the whole camera image as a representation of the place, most in a domain-transformed way, for instance, as PCA transformed image [159], Fourier-transformed image [213], image histograms, image fingerprints [182], GIST descriptors [253, 254], and so on. Local representations instead identify salient regions of the image first and create the representation out of this only. This approach largely depends on the detection of salient regions using interest point or interest region detectors, which we have seen in section 4.5. With the development of many effective interest point detectors, local methods have proven to be practical and are nowadays applied in many systems. We will therefore present this method first as the preferred way to location recognition. However, in the last two sections, we will also review some of the earliest approaches to place recognition using image histograms and fingerprints. In fact, although largely outperformed by the local *visual-word*-based approaches, these methods are still used in some robot applications.

4.6.2 From bag of features to visual words

A representation of an image by a set of interest points only is usually called a bag of features. For each interest, point a descriptor is usually computed in a manner that is invariant to rotation, scale, intensity, and viewpoint change (section 4.5.4). A popular way is to use gradient histograms, e.g. SIFT (section 4.5.5.1) or SURF (section 4.5.5.2). This set of descriptors is the new representation of the image. It is called a *bag of features* because the original spatial relation between the interest points is removed and only the descriptors are remembered. The similarity between two sets of descriptors can be computed by counting the number of common feature descriptors. For this, a matching function needs to be defined, which allows us to determine whether two feature descriptors are the same. This matching function usually depends on the type of feature descriptor. But in general a feature descriptor is a high-dimensional vector, and matching features can be found by computing the distance using the L_2 norm. *Visual words* are a 1-dimensional representation of the high-dimensional feature descriptor. This means that the visual word for a 128-dimensional SIFT descriptor is just a single integer number. The conversion to visual words creates a bag of visual words instead of a bag of features. For this conversion, the high-dimensional descriptor space is divided into nonoverlapping cells. This division is computed by *k-means clustering* [18]. For the clustering, a large number of feature descriptors is necessary. The computed cluster borders form the cell divisions of the feature space. Each of the cells is now assigned a number that will be assigned to any feature descriptor within the cell. This number is referred to as visual word. Similar feature descriptors will be then sorted into the same cell and therefore get the same visual word assigned. This is illustrated in figure 4.83, which is a very efficient method of finding matching-feature descriptors. The visual words created by the partitioning is called *visual vocabulary*.

For *quantization*, a prototype vector for each cell is stored, which is the mean descriptor vector of all training descriptors from the cell. To assign a feature descriptor to its cell it needs to be compared to all prototype vectors. For a large number of cells this can be a very expensive operation. It can be sped up by creating a hierarchical splitting of the feature space called *vocabulary tree* [243].

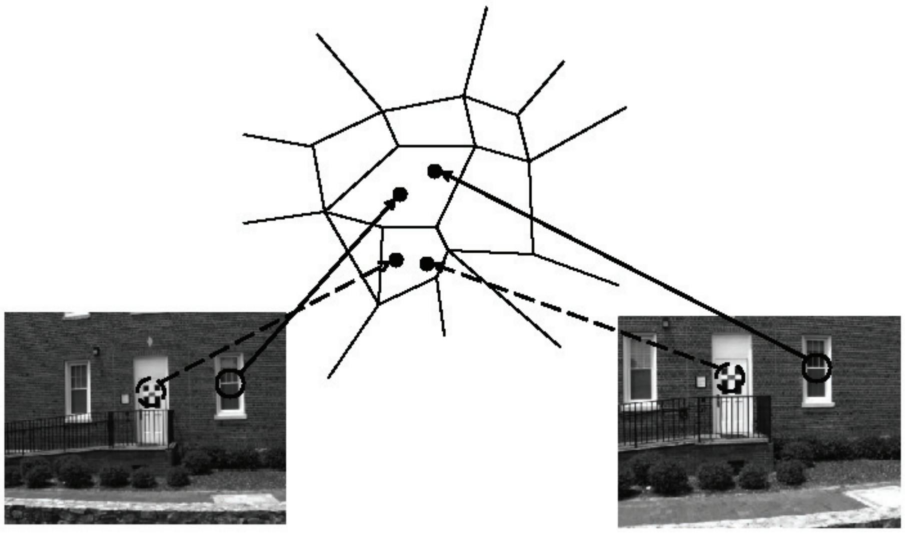


Figure 4.83 Partition of the descriptor feature space. Each cell stands for a visual word. Similar feature descriptors will be sorted into the same cell and therefore get the same visual word assigned.

4.6.3 Efficient location recognition by using an inverted file

Feature quantization into visual words is one key ingredient for efficient location recognition. Another one is the use of an *inverted file* for the database and a voting scheme for similarity computations. The database organized as an inverted file consists of a list of all possible visual words. Each element of this list points to another list that holds all the image identifiers in which this particular visual word appeared. This is illustrated in figure 4.84.

The voting scheme to find the most similar set of visual words in the database to a given query set works as follows. A voting array is initialized, which has as many cells as images in the database. A visual word from the query image is taken, and the list of image identifiers attached to this visual word is processed. For all the image identifiers in the list, a vote is cast by increasing the value at the corresponding position in the voting array. The most similar image to the query image is then the one with the highest vote. This voting scheme can exactly compute the L_2 norm if the descriptor vectors are correctly normalized [243].

This algorithm not only gives the most similar image in the database but also creates a ranking of all images in the database by similarity without any additional computational cost. This can be used to robustify place recognition.

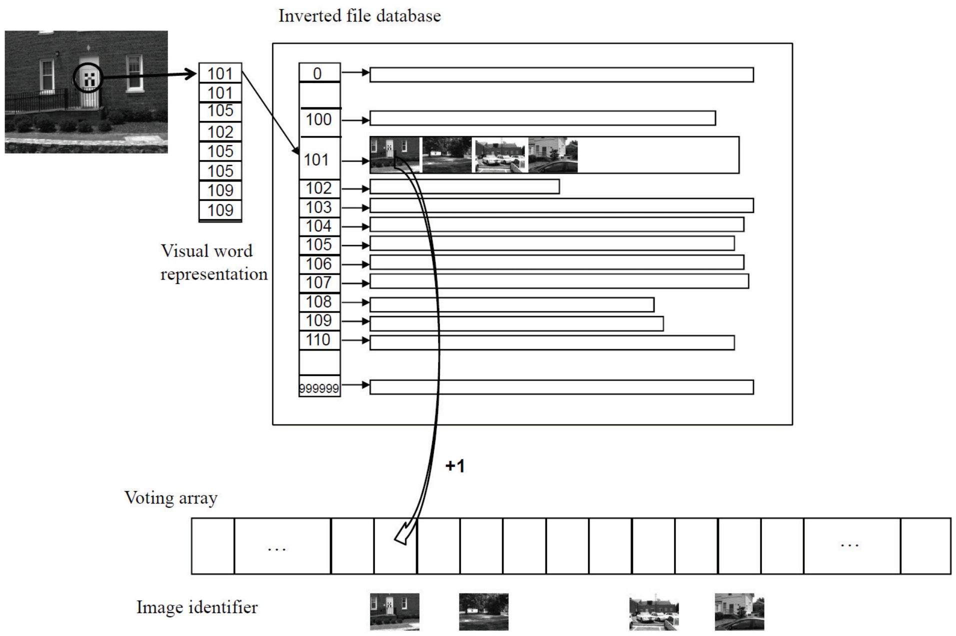


Figure 4.84 Visual word based location recognition using an inverted file system. An image in the database gets a vote if the same visual word is present in the query image.

4.6.4 Geometric verification for robust place recognition

The set of visual words does not contain the spatial relations anymore, thus an image that has the same visual words but in a different spatial arrangement would also have high similarity. The spatial relations however can be enforced again by a final geometric verification. For this, the k most similar images in a query are tested for geometric consistency. The geometric consistency test computes geometric transformations using the x and y image coordinates of matching visual words. Transformations used are affine transformations, homographies, or the essential matrix between images (section 4.2.6, page 184). The computation is performed in a robust way using RANSAC [128] (section 4.7.2.4), and the number of inliers to the transformation is counted. The image that achieves the largest number of inliers with the query image is then reported as the final match. This returns the desired location in place recognition.

4.6.5 Applications

This method for place recognition has already been used successfully in several applications. It was used for topological localization and mapping in [131]. It is also the core algorithm of FABMAP [108], for which it was extended by a probabilistic formulation. Other methods that use this scheme are described in [56, 276].

Available source code on the Web:

- Vocabulary-tree-based image search by F. Fraundorfer et al. [131]. It is a framework for fast image retrieval and place recognition very popular in robotics (useful for loop detection in visual SLAM):
  http://www1.ethz.ch/cvg/people/postgraduates/fraundof/vocsearch
- FABMAB, by M. Cummins et al. [108], is another framework for fast image retrieval and place recognition also very popular in robotics: http://www.robots.ox.ac.uk/~mobile/wikisite/pmwiki/pmwiki.php?n=Software.FABMAP
- Bag of features: another powerful tool for image retrieval and visual recognition using vocabulary trees: http://www.vlfeat.org/~vedaldi/code/bag/bag.html

These algorithms are all very useful for loop detection in the problem of simultaneous localization and mapping, which we will see in section 5.8.

4.6.6 Other image representations for place recognition

In this section, we review two of the early and most successful approaches to place recognition before the advent of the visual words based methods described above. The first method uses image histograms, while the second one uses image fingerprints.

4.6.6.1 Image histograms

A single visual image provides so much information regarding a robot's immediate surroundings that an alternative to searching the image for spatially localized features is to make use of the information captured by the entire image (i.e., all the image pixels) to extract a *whole-image feature* or *global image feature*. Whole-image features are not designed to identify specific spatial structures such as obstacles or the position of specific landmarks. Rather, they serve as compact representations of the entire local region. From the perspective of robot localization, the goal is to extract one or more features from the image that are correlated well with the robot's position. In other words, small changes in robot position should cause only small changes to whole-image features, while large changes in robot position should cause correspondingly large changes to whole-image features.

A logical first step in designing a vision sensor for this purpose is to maximize the field of view of the camera. As the field of view increases, a small-scale structure in the robot's environment occupies a smaller proportion of the image, thereby mitigating the impact of individual scene objects on image characteristics. A catadioptric camera system, nowadays very popular in mobile robotics, offers an extremely wide field of view (section 4.2.4).

A catadioptric image is a 360-degree image warped onto a 2D image surface. Because of this, it offers another critical advantage in terms of sensitivity to small-scale robot motion. If the camera is mounted vertically on the robot so that the image represents the environment surrounding the robot (i.e., its horizon; figure 4.37a), then rotation of the camera and robot simply results in image rotation. In short, the catadioptric camera can be invariant to rotation of the field of view.

Of course, mobile robot rotation will still change the image; that is, pixel positions will change, although the new image will simply be a rotation of the original image. But we intend to extract image features via histogramming. Because histogramming is a function of the set of pixel values and not of the position of each pixel, the process is pixel position-invariant. When combined with the catadioptric camera's field of view invariance, we can create a system that is invariant to robot rotation and insensitive to small-scale robot translation.

A color camera's output image generally contains useful information along multiple *bands*: r, g, and b values as well as hue, saturation, and luminance values. The simplest histogram extraction strategy is to build separate 1D histograms characterizing each band. Given a color camera image, G, the first step is to create mappings from G to each of the n available bands. We use G_i to refer to an array storing the values in band i for all pixels in G. Each band-specific histogram H_i is calculated as before:

- As preprocessing, smooth G_i using a Gaussian smoothing operator.
- Initialize H_i with n levels: $H[j] = 0$ for $j = 1, \ldots, n$.
- For every pixel (x,y) in G_i, increment the histogram: $H_i[G_i[x, y]] += 1$.

Given the image shown in figure 4.37a, the image histogram technique extracts six histograms (for each of r, g, b, hue, saturation, and luminance) as shown in figure 4.85. In order to make use of such histograms as whole-image features, we need ways to compare to histograms to quantify the likelihood that the histograms map to nearby robot positions. The problem of defining useful histogram distance metrics is itself an important subfield within the image retrieval field. For an overview refer to [270]. One of the most successful distance metrics encountered in mobile robot localization is the *Jeffrey divergence*. Given two histograms H and K, with h_i and k_i denoting the histogram entries, the Jeffrey divergence $d(H, K)$ is defined as

$$d(H, K) = \sum_i \left(h_i \log \frac{2h_i}{h_i + k_i} + k_i \log \frac{2k_i}{h_i + k_i} \right) . \tag{4.124}$$

Using measures such as the Jeffrey divergence, mobile robots have used whole-image histogram features to identify their position in real time against a database of previously

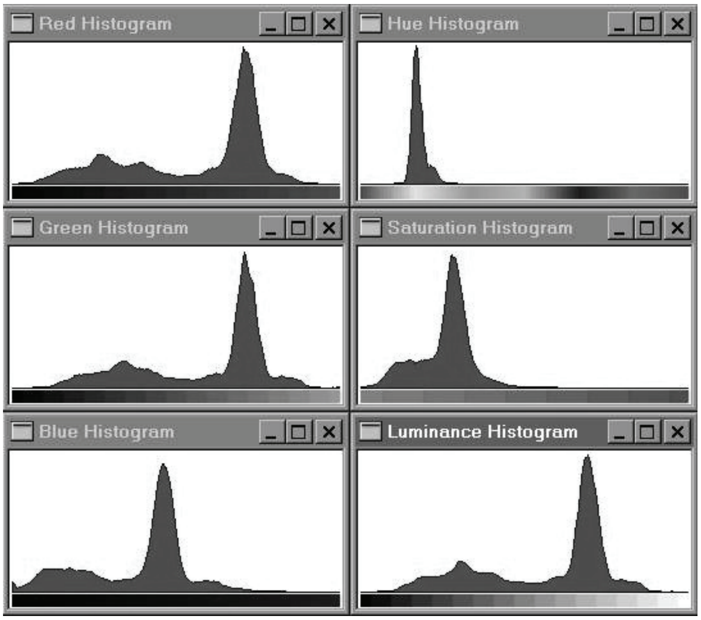


Figure 4.85
Six 1D histograms of the image above. A 5×5 smoothing filter was convolved with each band before histogramming.

recorded images of locations in their environment. Using this whole-image extraction approach, a robot can readily recover the particular hallway or particular room in which it is located [325].

Finally, note that in the last decade another global image descriptor—known as GIST—has been devised. The image is represented by a 320 dimensional vector per color band. The feature vector corresponds to the mean response to steerable filters at different scales and orientations computed over 4×4 sub-windows. Because a complete explanation of the GIST descriptor goes beyond the scope of this book, for an in-depth study we refer the reader to [253, 254] and [238].

4.6.6.2 Image fingerprints

This method is similar to the *visual word* approach with the difference that here the features are not interest points but rather morphological features, lines and colored blobs. Although outperformed by the new *visual word* based place recognition methods, this approach is still quite used in many applications of mobile robotics in both indoor and outdoor.

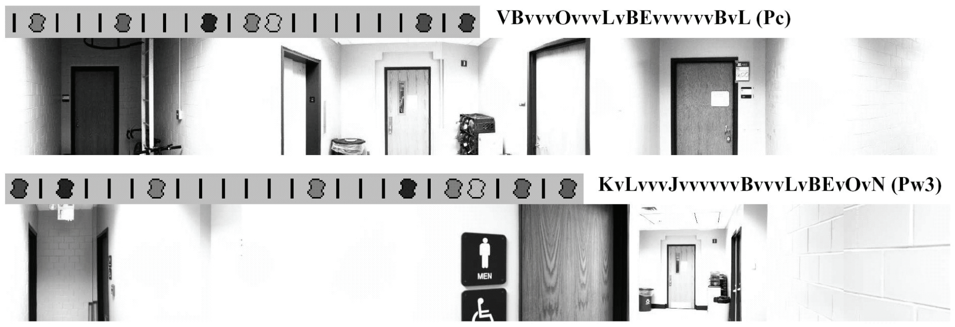


Figure 4.86
Two panoramic images and their associated fingerprint sequences [182].

We describe one particular implementation of this approach called the image *fingerprint*, which was developed first in [182]. Such as the previous method, the system makes use of a 360-degree panoramic image. The first extraction tier searches the panoramic image for spatially localized features: vertical edges and sixteen discrete hues of color. The vertical edge detector is a straightforward gradient approach implementing a horizontal difference operator. Vertical edges are "voted upon" by each edge pixel just as in a vertical edge Hough transform. An adaptive threshold is used to reduce the number of edges. Suppose the Hough table's tallies for each candidate vertical line have a mean μ and a standard deviation σ. The chosen threshold is simply $\mu + \sigma$.

Vertical color bands are identified in largely the same way, identifying statistics over the occurrence of each color, then filtering out all candidate color patches except those with tallies greater than $\mu + \sigma$. Figure 4.86 shows two sample panoramic images and their associated fingerprints. Note that each fingerprint is converted to an ASCII string representation.

Just as with histogram distance metrics in the case of image histogramming, we need a quantifiable measure of the distance between two fingerprint strings. String-matching algorithms are yet another large field of study, with particularly interesting applications today in the areas of genetics [55]. Note that we may have strings that differ not just in a single element value, but even in their overall length. For example, figure 4.87 depicts three actual sequences generated using the preceding algorithm. The top string should match *Place 1*, but note that there are deletions and insertions between the two strings.

The technique used in the fingerprinting approach for string differencing is known as a *minimum energy algorithm*. Taken from the stereo vision community, this optimization algorithm will find the minimum energy required to "transform" one sequence into another sequence. The result is a distance metric that is relatively insensitive to the addition or sub-

Bibliography

Books

[1] Adams, M.D., *Sensor Modelling: Design and Data Processing for Autonomous Navigation*. World Scientific Series in Robotics and Intelligent Systems. Singapore, World Scientific Publishing, 1999.

[2] Arkin, R.C., *Behavioral Robotics*. Cambridge MA, MIT Press, 1998.

[3] Bar-Shalom, Y., Li, X.-R., *Estimation and Tracking: Principles, Techniques, and Software.* Norwood, MA, Artech House, 1993.

[4] Benosman, R., Kang, S. B., *Panoramic Vision: Sensors, Theory, and Applications*, New York, Springer-Verlag, 2001.

[5] Borenstein, J., Everett, H.R., Feng, L., *Navigating Mobile Robots: Systems and Techniques*. Natick, MA, A.K. Peters, Ltd., 1996.

[6] Borenstein, J., Everett, H.R., Feng, L., *Where Am I? Sensors and Methods for Mobile Robot Positioning.* Technical report, Ann Arbor, University of Michigan, 1996. Available at http://www-personal.engin.umich.edu/~johannb/position.htm.

[7] Bradski, G., Kaehler, A., *Learning OpenCV: Computer Vision with the OpenCV Library*, Sebastopol, CA, O'Reilly Media, Inc., 1st edition, 2008.

[8] Breipohl, A.M., *Probabilistic Systems Analysis: An Introduction to Probabilistic Models, Decisions, and Applications of Random Processes.* New York, John Wiley & Sons, 1970.

[9] Bundy, A. (editor), *Artificial Intelligence Techniques: A Comprehensive Catalogue*. New York, Springer-Verlag, 1997.

[10] Canudas de Wit, C., Siciliano, B., and Bastin G. (editors), *Theory of Robot Control*. New York, Spinger, 1996.

[11] Carroll, R.J., Ruppert, D., *Transformation and Weighting in Regression*. New York, Chapman and Hall, 1988.

[12] Cox, I.J., Wilfong, G.T. (editors)*, Autonomous Robot Vehicles*. New York, Springer-Verlag, 1990.

[13] Craig, J.J., *Introduction to Robotics: Mechanics and Control*. 2nd edition. Boston, Addison-Wesley, 1989.

[14] De Silva, C.W., *Control Sensors and Actuators.* Upper Saddle River, NJ, Prentice-Hall, 1989.

[15] Daniillidis, K., Klette, R., *Imaging Beyond the Pinhole Camera*. New York, Springer, 2006.

[16] Dietrich, C.F., *Uncertainty, Calibration and Probability*. Bristol, UK, Adam Hilger, 1991.

[17] Draper, N.R., Smith, H., *Applied Regression Analysis*. 3rd edition. New York, John Wiley & Sons, 1988.

[18] Duda, R.O., Hart, P.E., Stork, D.G., *Pattern Classification*. New York, Wiley, 2001.

[19] Duda, R. O., Hart, P.E. *Pattern Classification and Scene Analysis*. New York, John Wiley & Sons, 1973.

[20] Everett, H.R., *Sensors for Mobile Robots: Theory and Applications*. New York, Natick, MA, A.K. Peters, Ltd., 1995.

[21] Faugeras, O., *Three-Dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, MA, MIT Press, 1993.

[22] Faugeras, O., Luong, Q.T., *The Geometry of Multiple Images*. Cambridge, MA, MIT Press, 2001.

[23] Floreano, D., Zufferey, J.C., Srinivasan, M.V., Ellington, C., *Flying Insects and Robots*, Springer, 2009.

[24] Forsyth, D. A., Ponce, J., *Computer Vision: A Modern Approach*. Upper Saddle River, NJ, Prentice Hall, 2003.

[25] Genesereth, M.R., Nilsson, N.J., *Logical Foundations of Artificial Intelligence.* Palo Alto, CA, Morgan Kaufmann, 1987.

[26] Gonzalez, R., Woods, R., *Digital Image Processing*. 3rd edition. New York, Pearson Prentice Hall, 2008.

[27] Hammond, J. H., *The Camera Obscura: A Chronicle*. Bristol, UK, Adam Hilger, 1981.

[28] Haralick, R.M., Shapiro, L.G., *Computer and Robot Vision, 1+2*. Boston, Addison-Wesley, 1993.

[29] Hartley, R.I., Zisserman, A. *Multiple View Geometry*. Cambridge, UK, Cambridge University Press, 2004.

[30] Jones, J., Flynn, A., *Mobile Robots, Inspiration to Implementation*. Natick, MA, A.K. Peters, Ltd., 1993.

[31] Kortenkamp, D., Bonasso, R.P., Murphy, R.R. (editors), *Artificial Intelligence and Mobile Robots; Case Studies of Successful Robot Systems*. Cambridge, MA, AAAI Press / MIT Press, 1998.

[32] Latombe, J.C., *Robot Motion Planning*. Norwood, MA, Kluwer Academic, 1991.

[33] LaValle, S.M. *Planning Algorithms*, Cambridge, UK, Cambridge University Press, 2006.

[34] Lee, D., *The Map-Building and Exploration Strategies of a Simple Sonar-Equipped Mobile Robot*. Cambridge, UK, Cambridge University Press, 1996.

[35] Leonard, J.E., Durrant-Whyte, H.F., *Directed Sonar Sensing for Mobile Robot Navigation*. Norwood, MA, Kluwer Academic, 1992.

[36] Ma, Y., S. Soatto, S., Kosecka, J., Sastry, S., *An Invitation to 3-D Vision: From Images to Geometric Models*. New York, Springer-Verlag, 2003.

[37] Manyika, J., Durrant-Whyte, H.F., *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Palo Alto, CA, Ellis Horwood, 1994.

[38] Mason, M., *Mechanics of Robotics Manipulation*. Cambridge, MA, MIT Press, 2001.

[39] Murphy, R.R., *Introduction to AI Robotics.* Cambridge, MA, MIT Press, 2000.

[40] Nourbakhsh, I., *Interleaving Planning and Execution for Autonomous Robots*. Norwood, MA, Kluwer Academic, 1997.

[41] Papoulis, A. *Probability, Random Variables, and Stochastic Processes*, 4th edition. New York, McGraw-Hill, 2001.

[42] Raibert, M.H., *Legged Robots That Balance*. Cambridge, MA, MIT Press, 1986.

[43] Ritter, G.X., Wilson, J.N., *Handbook of Computer Vision Algorithms in Image Algebra*. Boca Raton, FL, CRC Press, 1996.

[44] Russell, S., Norvig, P., *Artificial Intelligence: A Modern Approach*. 3rd edition. New York, Prentice Hall International, 2010.

[45] Schraft, R.D., Schmierer, G., *Service Roboter*. Natick, MA, A.K. Peters, Ltd, 2000.

[46] Sciavicco, L., Siciliano, B., *Modeling and Control of Robot Manipulators*. New York, McGraw-Hill, 1996.

[47] Siciliano, B., Khatib, O., *Springer Handbook of Robotics*, Springer, 2008.

[48] Slama, C.C., *Manual of Photogrammetry*. 4th edition. Falls Church VA, American Society of Photogrammetry,1980.

[49] Szeliski, R., *Computer Vision: Algorithms and Applications*, New York, Springer, 2010.

[50] Tennekes, H., *The Simple Science of Flight: From Insects to Jumbo Jets*. Cambridge, MA, MIT Press, 1996.

[51] Thrun, S., Burgard, W., Fox, D., *Probabilistic Robotics*. Cambridge, MA, MIT Press, 2005.

[52] Todd, D.J, *Walking Machines: An Introduction to Legged Robots*. London, Kogan Page Ltd, 1985.

[53] Trucco, E., Verri, A., *Introductory Techniques for 3-D Computer Vision*. New York, Prentice Hall, 1998.

[54] Zufferey, J.C., *Bio-inspired Flying Robots: Experimental Synthesis of Autonomous Indoor Flyers*, EPFL Press, 2008.

Papers

[55] Aho, A.V., “Algorithms for finding patterns in strings,” in J. van Leeuwen (editor), *Handbook of Theoretical Computer Science*, Cambridge, MA, MIT Press, 1990, Volume A, chapter 5, 255–300.

[56] Angeli, A., Filliat, D., Doncieux, S., Meyer, J.A., “Fast and incremental method for loop-closure detection using bags of visual words,” *IEEE Transactions on Robotics*, 24(5): 1027–1037, October, 2008.

[57] Arras, K.O., Castellanos, J.A., Siegwart, R., “Feature multi-hypothesis localization and tracking for mobile robots using geometric constraints,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA’2002)*, Washington, DC, May , 2002.

[58] Arras, K.O., Persson, J., Tomatis, N., Siegwart, R., “Real-time obstacle avoidance for polygonal robots with a reduced dynamic window,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002),* Washington, DC, May, 2002.

[59] Arras, K.O., Siegwart, R.Y., “Feature extraction and scene interpretation for map navigation and map building,” in *Proceedings of SPIE, Mobile Robotics XII*, 1997.

[60] Arras, K.O., Tomatis, N., “Improving robustness and precision in mobile robot localization by using laser range finding and monocular vision,” in *Proceedings of the Third European Workshop on Advanced Mobile Robots (Eurobot 99)*, Zurich, September, 1999.

[61] Astolfi, A., “Exponential stabilization of a mobile robot,” *in Proceedings of 3rd European Control Conference,* Rome, September, 1995.

[62] Bailey, T., Durrant-Whyte, H., “Simultaneous localization and mapping: Part II,” *IEEE Robotics and Automation Magazine*, 108–117, 2006.

[63] Bailey, T., “Mobile robot localisation and mapping in extensive outdoor environments,” Ph.D. thesis, University of Sydney, 2002.

[64] Baker, S., Nayar, S., “A theory of single-viewpoint catadioptric image formation,” *International Journal of Computer Vision* 35, no. 2: 175–196, 1999.

[65] Barnard, K., Cardei V., Funt, B., “A comparison of computational color constancy algorithms,” *IEEE Transactions on Image Processing* 11: 972–984, 2002.

[66] Barreto, J. P., Araujo, H., “Issues on the geometry of central catadioptric image formation. *International Conference on Computer Vision and Pattern Recognition* (CVPR), 2001.

[67] Barreto, J. P., Araujo, H., “Fitting conics to paracatadioptric projection of lines,” *Computer Vision and Image Understanding* 101(3): 151–165. March, 2006.

[68] Barreto, J. P., Araujo, H., “Geometric properties of central catadioptric line images and their application in calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8): 1237–1333, August 2005.

[69] Barron, J.L., Fleet, D.J., Beauchemin, S.S., “Performance of optical flow techniques,” *International Journal of Computer Vision*, 12: 43–77, 1994.

[70] Batavia, P., Nourbakhsh, I., “Path planning for the cye robot,” *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’00)*, Takamatsu, Japan, November 2000.

[71] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., “Speeded-up robust features (SURF),” *International Journal on Computer Vision and Image Understanding* 110, no. 3: 346–359, 2008.

[72] Besl, P., McKay, N., “A method for registration of 3-D shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* (PAMI) 14, no. 2: 239–256, February 1992.

[73] Bicchi, A., Marigo, A., Piccoli, B., “On the reachability of quantized control systems,” *IEEE Transactions on Automatic Control*,. 4, no. 47: 546–563, 2002.

[74] Biederman, I., “Recognition-by-components: A theory of human image understanding,” *Psychological Review*, 2, no. 94: 115–147, 1987.

[75] Blackwell, D., “Conditional expectation and unbiased sequential estimation,” *Annals of Mathematical Statistics* 18: 105–110, 1947.

[76] Blösch, M., Weiss, S., Scaramuzza, D., Siegwart, R., “Vision based MAV navigation in unknown and unstructured environments,” *IEEE International Conference on Robotics and Automation* (ICRA 2010), Anchorage, Alaska, May 2010.

[77] Borenstein, J., Koren, Y., “The vector field histogram – fast obstacle avoidance for mobile robots.” *IEEE Journal of Robotics and Automation* 7: 278–288, 1991.

[78] Borges, G. A., Aldon, M.-J., “Line Extraction in 2D Range Images for Mobile Robotics,” *Journal of Intelligent and Robotic Systems* 40: 267–297, 2004.

[79] Bosse, M., Newman, P., Leonard, J., Teller, S., “Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework,” *International Journal of Robotics Research* 23, no. 12: 1113–1139, 2004.

[80] Bosse, M., Rikoski, R., Leonard, J., Teller, S., “Vanishing points and 3d lines from omnidirectional video,” *International Conference on Image Processing*, 2002.

[81] Brock, O., Khatib, O., “High-speed navigation using the global dynamic window approach,” *in Proceeding of the IEEE International Conference on Robotics and Automation*, Detroit, May 1999.

[82] Brooks, R., “A robust layered control system for a mobile robot,” *IEEE Transactions of Robotics and Automation*, RA-2:14–23, March 1986.

[83] Brown, H.B., Zeglin, G.Z., “The bow leg hopping robot", *in Proceedings of the IEEE International Conference on Robotics and Automation*, Leuwen, Belgium, May 1998.

[84] Bruce, J., Balch,T., and Veloso, M., “Fast and inexpensive color image segmentation for interactive robots,” *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’00)*, Takamatsu, Japan, 2000.

[85] Burgard,W., Cremers, A., Fox, D., Hahnel, D., Lakemeyer, G., Schulz, D., Steiner, W., Thrun, S., “Experiences with an interactive museum tour-guide robot,” *Artificial Intelligence* 114: 1–53, 2000.

[86] Burgard, W., Derr, A., Fox, D., Cremers, A., “Integrating Global Position Estimation and Position Tracking for Mobile Robots: The Dynamic Markov Localization Approach,” *in Proceedings of the 1998 IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS’98)*, Victoria, Canada, October 1998.

[87] Burgard, W., Fox, D., Henning, D., “Fast grid-based position tracking for mobile robots,” *in Proceedings of the 21th German Conference on Artificial Intelligence (KI97)*, Freiburg, Germany, Springer-Verlag, 1997.

[88] Burgard, W., Fox, D., Jans, H., Matenar, C., Thrun, S., “sonar mapping of large-scale mobile robot environments using EM,” *in Proceedings of the International Conference on Machine Learning*, Bled, Slovenia, 1999.

[89] Cabani, C., Mac Lean, W. J., “Implementation of an affine-covariant feature detector in field-programmable gate arrays,” in *Proceedings of the International Conference on Computer Vision Systems*, 2007.

[90] Campion, G., Bastin, G., D’Andréa-Novel, B., “Structural properties and classification of kinematic and dynamic models of wheeled mobile robots.” *IEEE Transactions on Robotics and Automation* 12, no. 1: 47–62, 1996.

[91] Canny, J. F., “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 679–698, 1986.

[92] Canudas de Wit, C., Sordalen, O.J., “Exponential stabilization of mobile robots with nonholonomic constraints.” *IEEE Transactions on Robotics and Automation* 37: 1791–1797, 1993.

[93] Caprari, G., Estier, T., Siegwart, R., “Fascination of down scaling–alice the sugar cube robot.” *Journal of Micro-Mechatronics* 1: 177–189, 2002.

[94] Caprile, B., Torre, V., “Using vanishing points for camera calibration.” *International Journal of Computer Vision*. 4: 127–140, 1990.

[95] Castellanos, J.A., Tardos, J.D., Schmidt, G., "Building a global map of the environment of a mobile robot: The importance of correlations," in *Proceedings of the* 1997 *IEEE Conference on Robotics and Automation*, Albuquerque, NM, April 1997.

[96] Castellanos, J.A., Tardos, J.D., "Laser-based segmentation and localization for a mobile robot," in *Robotics and Manufacturing: Recent Trends in Research and Applications*, volume 6. ASME Press, 1996.

[97] Censi, A., Carpin, S., "HSM3D: Feature-less global 6DOF scan-matching in the hough/radon domain," *IEEE International Conference on Robotics and Automation* (ICRA), 2009.

[98] Chen, C.T., Quinn, R.D., "A crash avoidance system based upon the cockroach escape response circuit," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Albuquerque, NM, April 1997.

[99] Chenavier, F., Crowley, J.L., "Position estimation for a mobile robot using vision and odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France, May 1992.

[100] Cheeseman, P., Smith, P. "On the representation and estimation of spatial uncertainty," *International Journal of Robotics* 5: 56–68, 1986.

[101] Chomat, O., Colin deVerdiere, V., Hall, D., Crowley, J., "Local scale selection for gaussian based description techniques," in *Proceedings of the European Conference on Computer Vision*, Dublin, Ireland, 117–133, 2000.

[102] Chong, K.S., Kleeman, L., "Accurate odometry and error modelling for a mobile robot," *in Proceedings of the IEEE International Conference on Robotics and Automation*, Albuquerque, NM, April 1997.

[103] Choset, H., Walker, S., Eiamsa-Ard, K., Burdick, J., "Sensor exploration: Incremental construction of the hierarchical generalized voronoi graph." *The International Journal of Robotics Research* 19: 126–148, 2000.

[104] Collins, A. Ruina, R. Tedrake, M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science* 307, no. 5712: 1082 - 1085, 2005.

[105] Csorba, M. "Simultaneous localisation and map building," *Ph.D. thesis*, University of Oxford, Oxford, 1997.

[106] Cox, I.J., Leonard, J.J., "Modeling a dynamic environment using a bayesian multiple hypothesis approach," *Artificial Intelligence* 66: 311–44, 1994.

[107] Corke, P.I., Strelow, D., Singh, S., "Omnidirectional visual odometry for a planetary rover," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[108] Cummins, M., Newman, P., "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research* 27(6): 647–665, 2008.

[109] Cummins, M., Newman, P., "Highly scalable appearance-only SLAM – FAB-MAP 2.0," *In Robotics Science and Systems (RSS)*, Seattle, USA, June 2009.

[110] Davison, A.J., "Real-time simultaneous localisation and mapping with a single camera," *International Conference on Computer Vision*, 2003.

[111] Davison, A.J. "Active search for real-time vision," *In International Conference on Computer Vision*, 2005.

[112] Davison, A. J., Reid, I., Molton, N., Stasse, O., "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, no. 6, June, 2007.

[113] Dellaert, F. "Square root SAM," *Proceedings of the Robotics Science and Systems Conference*, 2005.

[114] Dijkstra, E.W. "A note on two problems in connexion with graphs," *Numerische Mathematik* 1: 269–271, 1959.

[115] Dowlingn, K., Guzikowski, R., Ladd, J., Pangels, H., Singh, S., Whittaker, W.L., "NAVLAB: An autonomous navigation testbed," *Technical report CMU-RI-TR-87-24, Robotics Institute*, Pittsburgh, Carnegie Mellon University, November 1987.

[116] Duckett, T., Marsland, S.,Shapiro, J. "Learning globally consistent maps by relaxation," *IEEE International Conference on Robotics and Automation*, 2000.

[117] Duckett, T., Marsland, S.,Shapiro, J. "Fast, on-line learning of globally consistent maps," *Autonomous Robots* 12, no. 3: 287–300, 2002.

[118] Dudek, G., Jenkin, M., "Inertial sensors, GPS, and odometry," *Springer Handbook of Robotics*, Springer, 2008.

[119] Dugan, B., "Vagabond: A demonstration of autonomous, robust outdoor navigation," *in Video Proceedings of the IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993.

[120] Durrant-Whyte, H., Bailey, T., "Simultaneous localization and mapping: Part I," *IEEE Robotics and Automation Magazine*, 99–108, 2006.

[121] Einsele, T., "Real-time self-localization in unknown indoor environments using a panorama laser range finder," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 697–702, 1997.

[122] Elfes, A., "Sonar real world mapping and navigation," in [12].

[123] Ens, J., Lawrence, P., "An investigation of methods for determining depth from focus." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15: 97–108, 1993.

[124] Espenschied, K.S., Quinn, R.D., "Biologically-inspired hexapod robot design and simulation," *in AIAA Conference on Intelligent Robots in Field, Factory, Service and Space*, Houston, Texas, March, 1994.

[125] Falcone, E., Gockley, R., Porter, E., Nourbakhsh, I., "The personal rover project: the comprehensive design of a domestic personal robot," *Robotics and Autonomous Systems, Special Issue on Socially Interactive Robots* 42: 245–258, 2003.

[126] Feder, H.J.S., Slotine, J-J.E., "Real-time path planning using harmonic potentials in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Albuquerque, NM, April 1997.

[127] Ferguson, D., Howard, T., Likhachev, M., "Motion planning in urban environments: Part II," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS), 2008.

[128] Fischler, M. A., Bolles, R. C. "RANSAC random sampling concensus: A paradigm for model fitting with applications to Image analysis and automated cartography,". *Communications of ACM* 26: 381–395, 1981.

[129] Fox, D., "KLD-sampling: Adaptive particle filters and mobile robot localization," *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[130] Fox, D., Burgard,W., Thrun, S., “The dynamic window approach to collision avoidance,” *IEEE Robotics and Automation Magazine* 4: 23–33, 1997.

[131] Fraundorfer, F., Engels, C., Nister, D., “Topological mapping, localization and navigation using image collections,” *IEEE/RSJ Conference on Intelligent Robots and Systems* 1, 2007.

[132] Freedman, B., Shpunt, A., Machline, M., Arieli, Y., “Depth mapping using projected patterns,” *US Patent no. US20100118123A1*, May 13, 2010. http://www.freepatentsonline.com/20100118123.pdf

[133] Fusiello, A., Trucco, E., Verri, A., “A compact algorithm for rectification of stereo pairs,” *Machine Vision and Applications*, 12(1): 16–22, 2000.

[134] Gächter, S., Harati, A., Siegwart, R., “Incremental object part detection toward object classification in a sequence of noisy range images,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2008)*, Pasadena, USA, May 2008.

[135] Gander,W., Golub, G.H., Strebel, R., “Least-squares fitting of circles and ellipses,” *BIT Numerical Mathematics* 34, no. 4: 558–578, December 1994.

[136] Genesereth, M.R. “Deliberate agents,” *Technical Report Logic-87-2.* Stanford, CA, Stanford University, Logic Group, 1987.

[137] Geyer, C., Daniilidis, K., “A unifying theory for central panoramic systems and practical applications,” *European Conference on Computer Vision* (ECCV), 2000.

[138] Goedeme, T., Nuttin, M., Tuytelaars, T., Van Gool, L., “Markerless computer vision based localization using automatically generated topological maps,” *European Navigation Conference* GNSS, Rotterdam, 2004.

[139] Golfarelli, M., Maio, D., Rizzi, S. “Elastic correction of dead-reckoning errors in map building,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[140] Golub, G., Kahan,W., “Calculating the singular values and pseudo-inverse of a matrix.” *Journal SIAM Numerical Analysis* 2: 205–223, 1965.

[141] Grisetti, G., Stachniss, C., Grzonka, S., Burgard, W., “A tree parameterization for efficiently computing maximum likelihood maps using gradient descent,” *Robotics Science and Systems* (RSS), 2007.

[142] Grzonka, S., Grisetti, G., Burgard, W. “Towards a navigation system for autonomous indoor flying,” *IEEE International Conference on Robotics and Automation*, 2009.

[143] Gutmann, J.S., Burgard, W., Fox, D., Konolige, K., “An experimental comparison of localization methods,” in *Proceedings of the 1998 IEEE/RSJ International. Conference of Intelligent Robots and Systems* (IROS’98), Victoria, Canada, October 1998.

[144] Guttman, J.S., Konolige, K., “Incremental mapping of large cyclic environments,” in P*roceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, Monterey, November 1999.

[145] Hähnel, D., Fox, D., Burgard, W., Thrun, S. “A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements,” *Proceedings of the Conference on Intelligent Robots and Systems*, 2003.

[146] Harris, C., Stephens, M., “A combined corner and edge detector,” *Proceedings of the 4th Alvey Vision Conference*, 1988.

[147] Hart, P. E., Nilsson, N. J., Raphael, B. “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions on Systems Science and Cybernetics* 4, no. 2: 100–107, 1968.

[148] Hashimoto, S., “Humanoid robots in Waseda University—Hadaly-2 and WABIAN,” *in IARP First International Workshop on Humanoid and Human Friendly Robotics*, Tsukuba, Japan, October 1998.

[149] Heale, A., Kleeman, L.: “A real time DSP sonar echo processor,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’00)*, Takamatsu, Japan, 2000.

[150] Heymann, S., Maller, K., Smolic, A., Froehlich, B., Wiegand, T., “SIFT implementation and optimization for general-purpose GPU,” *in Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2007.

[151] Horn, B.K.P., Schunck, B.G., “Determining optical flow,” *Artificial Intelligence*, 17: 185–203, 1981.

[152] Horswill, I., “Visual collision avoidance by segmentation,” in *Proceedings of IEEE International Conference on Robotics and Automation*, 902–909, 1995, IEEE Press, Munich, November 1994.

[153] Hoyt, D.F., Taylor, C.R, “Gait and the energetics of locomotion in horses,” *Nature* 292: 239–240, 1981.

[154] Jacobs, R. and Canny, J., “Planning smooth paths for mobile robots,” in *Proceeding. of the IEEE Conference on Robotics and Automation*, IEEE Press, 2–7, 1989.

[155] Jeffreys, H. and Jeffreys, B. S. “Methods of mathematical physics,” *Cambridge, Cambridge University Press*, 305-306, 1988.

[156] Jennings, J., Kirkwood-Watts, C., Tanis, C., “Distributed map-making and navigation in dynamic environments,” in *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’98)*, Victoria, Canada, October 1998.

[157] Jensfelt, P., Austin, D., Wijk, O., Andersson, M., “Feature based condensation for mobile robot localization,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, May 24–28, 2000.

[158] Jensfelt, P., Christensen, H., “Laser based position acquisition and tracking in an indoor environment,” in *Proceedings of the IEEE International Symposium on Robotics and Automation* 1, 1998.

[159] Jogan, M., Leonardis, A. “Robust localization using panoramic viewbased recognition,” in *Proceedings of ICPR00* 4: 136–139, 2000.

[160] Jung, I., Lacroix, S., “Simultaneous localization and mapping with stereovision,” in *Proceedings of the 11th International Symposium Robotics Research*, Siena, Italy, 2005.

[161] Kamon, I., Rivlin, E., Rimon, E., “A new range-sensor based globally convergent navigation algorithm for mobile robots,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, April 1996.

[162] Kelly, A., “Pose determination and tracking in image mosaic based vehicle position estimation,” in *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’00)*, Takamatsu, Japan, 2000.

[163] Khatib, O., Real-time obstacle avoidance for manipulators and mobile robots, *International Journal of Robotics Research* 5, no. 1, 1986.

[164] Khatib, M., Chatila, R., “An extended potential field approach for mobile robot sensor motions,” in *Proceedings of the Intelligent Autonomous Systems IAS-4*, IOS Press, Karlsruhe, Germany, March 1995, 490–496.

[165] Khatib, M., Jaouni, H., Chatila, R., Laumod, J.P., “Dynamic path modification for car-like nonholonomic mobile robots,” in *Proceedings of IEEE International Conference on Robotics and Automation*, Albuquerque, NM, April 1997.

[166] Khatib, O., Quinlan, S., “Elastic bands: connecting, path planning and control,” in *Proceedings of IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993.

[167] Klein, G., Murray, D., “Parallel Tracking and Mapping for Small AR Workspaces,” *Proceedings of the International Symposium on Mixed and Augmented Reality* (ISMAR'07), Nara, Japan, 2007.

[168] Ko, N.Y., Simmons, R., “The lane-curvature method for local obstacle avoidance,” in *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’98)*, Victoria, Canada, October 1998.

[169] Koenig, S., Simmons, R., “Xavier: A robot navigation architecture based on partially observable markov decision process models,” in [31].

[170] Koenig, S., Likhachev, M., “Fast replanning for navigation in unknown terrain,” *IEEE Transactions on Robotics* 21(3): 354–363, 2005.

[171] Konolige, K.,. “A gradient method for realtime robot control,” in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, Takamatsu, Japan, 2000.

[172] Konolige, K., “Small vision systems: Hardware and implementation,” in P*roceedings of Eighth International Symposium on Robotics Research*, Hayama, Japan, October 1997.

[173] Konolige, K., “Large-scale map-making,” *AAAI National Conference on Artificial Intelligence*, 2004.

[174] Konolige, K., Agrawal, M., Solà, J., “Large scale visual odometry for rough terrain,” *International Symposium on Research in Robotics* (ISRR), November, 2007.

[175] Koperski, K., Adhikary, J., Han, J., “Spatial data mining: Progress and challenges survey paper,” in *Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, June 1996.

[176] Koren, Y., Borenstein, J., “High-speed obstacle avoidance for mobile robotics,” in *Proceedings of the IEEE Symposium on Intelligent Control* 382–384, Arlington, VA, August 1988.

[177] Koren, Y., Borenstein, J., “Real-time obstacle avoidance for fast mobile robots in cluttered environments,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, Los Alamitos, CA, May 1990.

[178] Kruppa, E., “Zur ermittlung eines objektes aus zwei perspektiven mit innerer orientierung,” *Sitzungsberichte Österreichische Akademie der Wissenschaften, Mathematisch-naturwissenschaftliche Klasse, Abteilung II* a, volume 122: 1939-1948, 1913.

[179] Kuipers, B., Byun, Y.T., "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Journal of Robotics and Autonomous Systems*, 8: 47–63, 1991.

[180] Kuo, A., "Choosing your steps carfully," *Robotics & Automation Magazine*, 2007.

[181] Lacroix, S., Mallet, A., Chatila, R., Gallo, L., "Rover self localization in planetary-like environments," *in Proc. Int. Symp. Artic. Intell., Robot., Autom. Space* (i-SAIRAS), Noordwijk, The Netherlands, 1999.

[182] Lamon, P., Nourbakhsh, I., Jensen, B., Siegwar,t R., "Deriving and matching image fingerprint sequences for mobile robot localization," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001.

[183] Latombe, J.C., Barraquand, J., "Robot motion planning: A distributed presentation approach." *International Journal of Robotics Research*, 10: 628–649, 1991.

[184] Lauria, M., Estier, T., Siegwart, R.: "An innovative space rover with extended climbing abilities," in *Video Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, May 2000.

[185] LaValle, S. M., "Rapidly-exploring random trees: A new tool for path planning," *Technical Report, Computer Science Dept*., Iowa State University, October 1998.

[186] Lavalle, S. M.: "Rapidly-exploring random trees: Progress and prospects," In *Algorithmic and Computational Robotics: New Directions*, pp. 293-308, 2000.

[187] Lazanas, A., Latombe, J.C., "Landmark robot navigation," in *Proceedings of the Tenth National Conference on AI*. San Jose, CA, July 1992.

[188] Lazanas, A. Latombe, J.C., "Motion planning with uncertainty: A landmark approach." *Artificial Intelligence*, 76: 285–317, 1995.

[189] Lee, S.-O., Cho, Y.-J., Hwang-Bo, M., You, B.-J., Oh, S.-R.: "A stabile target-tracking control for unicycle mobile robots," in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, 2000.

[190] Leonard, J.J., Rikoski, R.J., Newman, P.M., Bosse, M., "Mapping partially observable features from multiple uncertain vantage points," *International Journal of Robotics Research* 21, no. 10: 943–975, 2002.

[191] Likhachev, M., Gordon, G., Thrun, S. "ARA*: Anytime A* with provable bounds on sub-optimality," *Advances in Neural Information Processing Systems* (NIPS), 2003.

[192] Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., Thrun, S., "Anytime dynamic A*: An anytime, replanning algorithm," *Proceedings of the International Conference on Automated Planning and Scheduling* (ICAPS), 2005.

[193] Lindeberg, T., "Feature detection with automatic scale selection," *International Journal of Computer Vision* 30, no. 2: 79-116, 1998.

[194] Longuet-Higgins, H.C., "A computer algorithm for reconstructing a scene from two projections," *Nature* 293: 133–135, September, 1981.

[195] Louste, C. and Liegois, A., Path planning for non-holonomic vehicles: a potential viscous fluid method, Robotica 20: 291–298, 2002.

[196] Lowe, David G., "Object recognition from local scale-invariant features," *Proceedings of the International Conference on Computer Vision,* 1999.

[197] Lowe, D. G., "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision* 60 (2): 91-110, 2004.

[198] Lumelsky, V., Skewis, T., "Incorporating range sensing in the robot navigation function," *IEEE Transactions on Systems, Man, and Cybernetics* 20: 1058–1068, 1990.

[199] Lumelsky, V., Stepanov, A., "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," in [12].

[200] Lu, F., Milios, E. "Globally consistent range scan alignment for environment mapping," *Autonomous Robots* 4: 333–349,1997.

[201] Maes, P., "The dynamics of action selection," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, 1989.

[202] Maes, P., "Situated Agents Can Have Goals," *Robotics and Autonomous Systems*, 6: 49–70. 1990.

[203] Maimone, M., Cheng, Y., Matthies, L., "Two years of visual odometry on the mars exploration rovers," *Journal on Field Robotics* 24, no. 3: 169–186, 2007.

[204] Makadia, A., Patterson, A., Daniilidis, K., "Fully automatic registration of 3D point clouds," *IEEE Conference on Computer Vision and Pattern Recognition*, New York, June 2006.

[205] Martinelli, A., Siegwart, R., "Estimating the odometry error of a mobile robot during navigation," in *Proceedings of the European Conference on Mobile Robots (ECMR 2003)*, Warsaw, September 4–6, 2003.

[206] Masoud, S.A., Masoud, A.A., "Motion planning in the presence of directional and regional avoidance constraints unsing nonlinear, anisotropic, harmonic potential fields: a physical metaphor," *IEEE Transactions on Systems, Man and Cybernetics* 32, no. 6: 705–723, 2002.

[207] Masoud, S.A., Masoud, A.A., "Kinodynamic motion planning: a novel type of nonlinear, passive damping forces and advantages," *IEEE Robotics Automation Magazine* 17, no. 1: 85–99, 2010.

[208] Matsumoto, Y., Inaba, M., Inoue, H., "Visual navigation using viewsequenced route representation," *IEEE International Conference on Robotics and Automation*, 1996.

[209] Maybeck,P.S., "The Kalman filter: An introduction to concepts," in [12].

[210] Matas, J., Chum, O., Urban, M., Pajdla, T., "Robust wide-baseline stereo from maximally stable extremal regions," in *Proceedings of the British Machine Vision Conference*, 384–393, 2002.

[211] McGeer, T., "Passive dynamic walking," *International Journal of Robotics Research* 9, no. 2: 62–82, 1990.

[212] Mei, C., Rives, P., "Single view point omnidirectional camera calibration from planar grids," *IEEE International Conference on Robotics and Automation* (ICRA), 2007.

[213] Menegatti, E., Maedab, T., Ishiguro, H., "Image-based memory for robot navigation using properties of omnidirectional images," *Robotics and Autonomous System* 47, no. 4: 251–267, July, 2004.

[214] Meng, M., Kak, A.C.. "Mobile robot navigation using neural networks and nonmetrical environmental models," *IEEE Control Systems Magazine*, 13(5): 30–39, October 1993.

[215] Metropolis, N., Ulam, S. "The Monte Carlo method," *Journal of the American Stattistical Association* 44, no. 247: 335–341, 1949.

[216] Mikolajczyk, K., C. Schmid, "Indexing based on scale-invariant interest points," *in Proceedings of the International Conference on Computer Vision*, 525–531, Vancouver, Canada, 2001.

[217] Mikolajczyk, K., Schmid, C., "Scale and affine invariant interest point detectors," *International Journal of Computer Vision* 1, no. 60: 63–86, 2004.

[218] Mikolajcyk, K. and Schmid, C., "An affine invariant interest point detector," *in Proceedings of the 7th European Conference on Computer Vision*, Denmark, 2002.

[219] Mikolajczyk, K., "Scale and Affine Invariant Interest Point Detectors," PhD thesis, INRIA Grenoble, 2002.

[220] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T.,Van Gool, L. "A comparison of affine region detectors," *International Journal of Computer Vision*, 65(1-2): 43–72, 2005.

[221] Minetti, A.E. ,Ardigò, L.P., Reinach, E., Saibene, F., "The relationship between mechanical work and energy expenditure of locomotion in horses," *Journal of Experimental Biology* 202, no. 17, 1999.

[222] Minguez, J., Montano, L., "Nearness diagram navigation (ND): A new real time collision avoidance approach," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, October 2000.

[223] Minguez, J., Montano, L., "Robot navigation in very complex, dense, and cluttered indoor / outdoor environments," in *Proceeding of International Federation of Automatic Control (IFAC2002)*, Barcelona, April 2002.

[224] Minguez, J., Montano, L., Khatib, O., "Reactive collision avoidance for navigation with dynamic constraints," in *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[225] Minguez, J., Montano, L., Simeon, T., Alami, R., "Global nearness diagram navigation (GND)," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 2001.

[226] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D. and Martinoli, A. "The e-puck, a robot designed for education in engineering," *The 9th Conference on Autonomous Robot Systems and Competitions*, 2009.

[227] Montiel, J.M.M. , Civera, J., Davison, A.J., "Unified inverse depth parametrization for monocular SLAM," *Proc. of the Robotics Science and Systems Conference*, 2006.

[228] Moutarlier, P., Chatila, R., "An experimental system for incremental environment modeling by an autonomous mobile robot," *1st International Symposium on Experimental Robotics*, 1989.

[229] Moutarlier, P., Chatila, R. "Stochastic multisensory data fusion for mobile robot location and environment modeling," *5th Int. Symposium on Robotics Research*, 1989.

[230] Montano, L., Asensio, J.R., "Real-time robot navigation in unstructured environments using a 3D laser range finder," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot and Systems*, IROS 97, September 1997.

[231] Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B. "FastSLAM: A factored solution to the simultaneous localization and mapping problem," *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002.

[232] Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B. “Fast-SLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges,” *International Joint Conference on Artificial Intelligence*, 2003.

[233] Moravec, H. and Elfes, A.E., “High Resolution Maps from Wide Angle Sonar,” in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, March 1985.

[234] Moravec, H. P., “Towards automatic visual obstacle avoidance,” *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, 1977.

[235] Moravec, H. P., “Visual mapping by a robot rover,” *International Joint Conference on Artificial Intelligence*, 1979.

[236] Moravec, H., “Obstacle avoidance and navigation in the real world by a seeing robot rover,” *PhD thesis*, Stanford University, 1980.

[237] Moutarlier, P., Chatila, R., “Stochastic multisensory data fusion for mobile robot location and environment modelling,” in *Proceedings of the 5th International Symposium of Robotics Research*, Tokyo, 1989.

[238] Murillo, A.C., Kosecka, J., “Experiments in Place Recognition using Gist Panoramas,” *Proceedings of the International Workshop on Omnidirectional Vision* (OMNIVIS’09), 2009.

[239] Murphy, K., Russell, S. “Rao-Blackwellized particle filtering for dynamic Bayesian networks,” *In Sequential Monte Carlo Methods in Practice*, ed. by A. Doucet, N. de Freitas, N. Gordon, 499–516, Springer, 2001.

[240] Nayar, S.K., “Catadioptric omnidirectional camera.” *IEEE CVPR*, 482–488, 1997.

[241] Nayar, S., Watanabe, M., and Noguchi, M., “Real-time focus range sensor.” *In Fifth International Conference on Computer Vision*, 995–1001, Cambridge, Massachusetts, 1995.

[242] Nilsson, N.J., “Shakey the robot.” *SRI, International, Technical Note*, Menlo Park, CA, 1984, No. 325.

[243] Nistér, D. Stewénius, H., “Scalable recognition with a vocabulary tree,” *IEEE International Conference on Computer Vision and Pattern Recognition*, 2006.

[244] Nistér, D., Naroditsky, O., Bergen, J., “Visual odometry for ground vehicle applications,” *Journal of Field Robotics* 23, no. 1: 3–20, 2006.

[245] Nistér, D., Naroditsky, O., Bergen, J., “Visual odometry,” *IEEE International Conference on Computer Vision and Pattern Recognition*, 2004.

[246] Nistér, D., “An efficient solution to the five-point relative pose problem,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* (PAMI), 26(6): 756-770, June 2004.

[247] Nguyen, V., Martinelli, A., Tomatis, N., Siegwart, R. “A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics,” *IEEE/RSJ Intenational Conference on Intelligent Robots and Systems*, IROS, 2005.

[248] Noth, André, “Design of solar powered airplanes for continuous flight,” *Ph.D. thesis, Autonomous Systems Lab, ETH Zurich*, Switzerland, December 2008.

[249] Nourbakhsh, I.R., “Dervish: An office-navigation robot,” in [31].

[250] Nourbakhsh, I.R., Andre. D., Tomasi, C., Genesereth, M.R., “Mobile robot obstacle avoidance via depth from focus,” *Robotics and Autonomous Systems*, 22: 151–158, 1997.

[251] Nourbakhsh, I.R., Bobenage, J., Grange, S., Lutz, R., Meyer, R, Soto, A., "An affective mobile educator with a full-time job," *Artificial Intelligence*, 114: 95–124, 1999.

[252] Nourbakhsh, I.R., Powers, R., Birchfield, S., "DERVISH, an office-navigation robot." *AI Magazine*, 16: 39–51, summer 1995.

[253] Oliva, A., Torralba, A., "Modeling the shape of the scene: A holistic representation of the spatial envelope," International Journal of Computer Vision, 42(3):145–175, 2001.

[254] Oliva, A., Torralba, A., "Building the gist of a scene: The role of global image features in recognition," in *Visual Perception, Progress in Brain Research*, 155:23–36, Elsevier, 2006.

[255] Omer, A.M.M., Ghorbani, R., Hun-ok Lim, Takanishi, A., "Semi-passive dynamic walking for biped walking robot using controllable joint stiffness based on dynamic simulation," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Singapore, 2009.

[256] Pell, B., Bernard, D., Chien, S., Gat, E., Muscettola, N., Nayak, P., Wagner, M., Williams, B., "An autonomous spacecraft agent prototype," *Autonomous Robots* 5: 1–27, 1998.

[257] Pavlidis, T., Horowitz, S. L. "Segmentation of plane curves," *IEEE Transactions on Computers* C-23(8): 860–870, 1974.

[258] Pentland, A.P., "A new sense for depth of field," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI),* 9: 523–531, 1987.

[259] Philippsen, R., Siegwart, R., "Smooth and efficient obstacle avoidance for a tour guide robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2003)*, Taipei, Taiwan, 2003.

[260] Pivtoraiko, M., Knepper, R., A., Kelly, A. "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics* 26, no. 1: 308–333, 2009.

[261] Pfister, S. T., Roumeliotis, S. I., Burdick, J. W. "Weighted line fitting algorithms for mobile robot map building and efficient data representation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.

[262] Pratt, J., Pratt, G., "Intuitive control of a planar bipedal walking robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '98)*, Leuven, Belgium, May 1998.

[263] Rao, C.R."Information and accuracy obtainable in estimation of statistical parameters," *Bulletin of the Calcutta Mathematical Society* 37: 81–91, 1945.

[264] Raibert, M. H., Brown, H. B., Jr., Chepponis, M., "Experiments in balance with a 3D one-legged hopping machine," *International Journal of Robotics Research*, 3: 75–92, 1984.

[265] Remy, C., Buffinton, K., Siegwart, R., "Stability analysis of passive dynamic walking of quadrupeds," *International Journal of Robotics Research*, 2009.

[266] Ringrose, R., "Self-stabilizing running," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '97)*, Albuquerque, NM, April 1997.

[267] Rosten, E., Drummond, T., "Fusing points and lines for high performance tracking," in *Proceedings of the International Conference on Computer Vision*, 1508–1511, 2005.

[268] Rosten, E., Drummond, T., “Machine learning for high-speed corner detection,” in *Proceedings of the European Conference on Computer Vision*, 430-443, 2006.

[269] Rowe, A., Rosenberg, C., Nourbakhsh, I., “A simple low cost color vision system,” in *Proceedings of Tech Sketches for CVPR 2001*, Kuaii, Hawaii, December 2001.

[270] Rubner, Y., Tomasi, C., Guibas, L., “The earth mover’s distance as a metric for image retrieval,” *STAN-CS-TN-98-86, Stanford University*, 1998.

[271] Rufli, M., Ferguson, D., Siegwart, R., “Smooth path planning in constrained environments,” *Proceedings of the IEEE International Conference on Robotics and Automation* (ICRA), 2009.

[272] Rufli, M., Siegwart, R., “On the application of the D* search algorithm to time based planning on lattice graphs,” *Proceedings of the European Conference on Mobile Robots* (ECMR), 2009.

[273] Scaramuzza, D., “Omnidirectional vision: from calibration to robot motion estimation,”, *PhD thesis n. 17635, ETH Zurich*, February 2008.

[274] Scaramuzza, D., Martinelli, A., Siegwart, R., “A flexible technique for accurate omnidirectional camera calibration and structure from motion,” *IEEE International Conference on Computer Vision Systems (ICVS 2006)*, New York, January 2006.

[275] Scaramuzza, D., Martinelli, A. Siegwart, R., “A toolbox for easily calibrating omnidirectional cameras,” *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS 2006), Beijing, China, October 2006.

[276] Scaramuzza, D., Fraundorfer, F., Pollefeys, M., “Closing the loop in appearance-guided omnidirectional visual odometry by using vocabulary trees,” *Robotics and Autonomous System Journal (Elsevier)*, 2010.

[277] Scaramuzza, D., Fraundorfer, F., Pollefeys, M., and Siegwart, R., “Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints,” *IEEE International Conference on Computer Vision* (ICCV 2009), Kyoto, October, 2009.

[278] Scaramuzza, D., Fraundorfer, F., and Siegwart, R., Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC, *IEEE International Conference on Robotics and Automation* (ICRA 2009), Kobe, Japan, May 2009.

[279] Scaramuzza, D., Siegwart, R., “Appearance guided monocular omnidirectional visual odometry for outdoor ground vehicles,” *IEEE Transactions on Robotics* 24, no. 5, October 2008.

[280] Schlegel, C., “Fast local obstacle under kinematic and dynamic constraints,” in *Proceedings of the IEEE International Conference on Intelligent Robot and Systems (IROS 98)*, Victoria, Canada 1998.

[281] Schultz, A., Adams, W., “Continuous localization using evidence grids,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA’98)*, May 1998.

[282] Schweitzer, G., Werder, M., “ROBOTRAC – a mobile manipulator platform for rough terrain,” in *Proceedings of the International Symposium on Advanced Robot Technology (ISART),* Tokyo, Japan, March, 1991.

[283] Shi, J., Malik, J., “Normalized cuts and image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 82: 888–905, 2000.

[284] Shi, J., Tomasi, C., “Good features to track,” *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[285] Schmid, C., Mohr, R., Bauckhage, C., "Evaluation of interest point detectors," *International Journal of Computer Vision* 37, no. 2: 151–172, 2000.

[286] Se, S., Barfoot, T., Jasiobedzki, P., "Visual motion estimation and terrain modeling for planetary rovers," *Proceedings of the International Symposium on Artificial Intelligence for Robotics and Automation in Space*, 2005.

[287] Siadat, A., Kaske, A., Klausmann, S., Dufaut, M., Husson, R. "An optimized segmentation method for a 2D laser-scanner applied to mobile robot navigation," *Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications*, 1997.

[288] Siegwart R., Arras, K., Bouabdallah, S., Burnier, D., Froidevaux, G., Greppin, X., Jensen, B., Lorotte, A., Mayor, L., Meisser, M., Philippsen, R., Piguet, R., Ramel, G., Terrien, G., Tomatis, N., "Robox at Expo.02: A large scale installation of personal robots," *Journal of Robotics and Autonomous Systems* 42: 203–222, 2003.

[289] Siegwart, R., Lamon, P., Estier, T., Lauria, M, Piguet, R., "Innovative design for wheeled locomotion in rough terrain," *Journal of Robotics and Autonomous Systems* 40: 151–162, 2002.

[290] Simhon, S., Dudek, G., "A global topological map formed by local metric maps," *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, Victoria, Canada, October 1998.

[291] Simmons, R., "The curvature velocity method for local obstacle avoidance," *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, April 1996.

[292] Sinha, S. N., Frahm, J. M., Pollefeys, M., Genc, Y., "GPU video feature tracking and matching," *in EDGE, Workshop on Edge Computing Using New Commodity Architectures*, 2006.

[293] Sivic, J. and Zisserman, A., "Video Google: A text retrieval approach to object matching in videos," *Proceedings of the International Conference on Computer Vision*, 2003.

[294] Smith, R., Self, M., Cheeseman, P., "Estimating uncertain spatial relationships in robotics," *Autonomous Robot Vehicles,* I. J. Cox and G. T. Wilfong (editors), Springer-Verlag, 167–193, 1990.

[295] Smith, R.C. , Cheeseman, P., "On the representation and estimation of spatial uncertainty, *International Journal of Robotics Research* 5, no. 4: 56–68, 1986.

[296] Smith, S. M., Brady, J. M., "SUSAN - A new approach to low level image processing," *International Journal of Computer Vision* 23, no. 34: 45–78, 1997.

[297] Snavely, N., Seitz, S.M., Szeliski, R., "Photo Tourism: Exploring photo collections in 3D," *ACM Transactions on Graphics*, 25(3), August 2006.

[298] Snavely, N., Seitz, S.M., Szeliski, R., "Modeling the World from Internet Photo Collections," *International Journal of Computer Vision*, 2007

[299] Soatto, S., Brockett, R., "Optimal structure from motion: Local ambiguities and global estimates,", *International Conference on Computer Vision and Pattern Recognition*, 1998.

[300] Sordalen, O.J., Canudas de Wi,t C., "Exponential control law for a mobile robot: extension to path following," *IEEE Transactions on Robotics and Automation*, 9: 837–842, 1993.

[301] Sorg, H.W., “From serson to draper – two centuries of gyroscopic development,” *Navigation* 23: 313–324, 1976.

[302] Steinmetz, B.M., Arbter, K., Brunner, B., Landzettel, K., “Autonomous vision navigation of the nanokhod rover,” *Proceedings of i-SAIRAS 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2001.

[303] Stentz, A., “The focussed D* algorithm for real-time replanning,” in *Proceedings of IJCAI-95*, August 1995.

[304] Stentz, A., “Optimal and efficient path planning for partially-known environments,” *Proceedings of the International Conference on Robotics and Automation*, 1994.

[305] Stevens, B.S., Clavel, R., Rey, L., “The DELTA parallel structured robot, yet more performant through direct drive,” *Proceedings of the 23rd International Symposium on Industrial Robots*, 1992.

[306] Takeda, H., Facchinetti, C., Latombe, J.C., “Planning the motions of a mobile robot in a sensory uncertainty field,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16: 1002–1017, 1994.

[307] Tardif, J., Pavlidis, Y., Daniilidis, K., “Monocular visual odometry in urban environments using an omnidirectional camera,” *IEEE/RSJ International Confrence on Intelligent Robots and Systems*, 2008.

[308] Taylor, R., Probert, P., “Range finding and feature extraction by segmentation of images for mobile robot navigation,” *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA, 1996.

[309] Thrun, S., Burgard, W., Fox, D., “A probabilistic approach to concurrent mapping and localization for mobile robots.” *Autonomous Robots* 31: 1–25. 1998.

[310] Thrun, S., et al., “Minerva: A second generation museum tour-guide robot,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'99)*, Detroit, May 1999.

[311] Thrun, S., Fox, D., Burgard, W., Dellaert, F., “Robust Monte Carlo localization for mobile robots,” *Artificial Intelligence*, 128: 99–141, 2001.

[312] Thrun, S. “A probabilistic online mapping algorithm for teams of mobile robots,” *International Journal of Robotics Research* 20, no. 5: 335–363, 2001.

[313] Thrun, S. “Simultaneous localization and mapping,” *Springer Tracts in Advanced Robotics* 38, no. 5: 13–41, 2008.

[314] Thrun, S., Gutmann, J.-S., Fox, D., Burgard, W., Kuipers, B., “Integrating topological and metric maps for mobile robot navigation: A statistical approach,” *Proceedings of the National Conference on Artificial Intelligence (AAAI)*,1998.

[315] Thrun, S., Thayer, S., Whittaker, W., Baker, C., Burgard, W., Ferguson, D., Hähnel, D., Montemerlo, M., Morris, A., Omohundro, Z., Reverte, C., Whittaker, W. “Autonomous exploration and mapping of abandoned mines,” *IEEE Robotics and Automation Mag*azine 11, no. 4: 79–91, 2004.

[316] Tomasi, C., Shi, J., “Image deformations are better than optical flow,” *Mathematical and Computer Modelling* 24: 165–175, 1996.

[317] Tomatis, N., Nourbakhsh, I., Siegwart, R., “Hybrid simultaneous localization and map building: A natural integration of topological and metric,” *Robotics and Autonomous Systems* 44, 3–14, 2003.

[318] Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A., “Bundle adjustment — a modern synthesis,” *International Conference on Computer Vision*, 1999.

[319] Tsai, R. “A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses,” *IEEE Journal of Robotics and Automation* 3, no. 4: 323–344, August 1987.

[320] Tuytelaars, T., Mikolajczyk, K., “Local invariant feature detectors: a survey,” *Source, Foundations and Trends in Computer Graphics and Vision* 3 , no. 3, 2007.

[321] Tzafestas, C.S., Tzafestas, S.G., “Recent algorithms for fuzzy and neurofuzzy path planning and navigation of autonomous mobile robots,” *Systems-Science* 25: 25–39, 1999.

[322] Ulrich, I., Borenstein, J., “VFH*: Local obstacle avoidance with look-ahead verification,” *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, May 2000.

[323] Ulrich, I., Borenstein, J., “VFH+: Reliable obstacle avoidance for fast mobile robots,” *Proceedings of the International Conference on Robotics and Automation (ICRA’98)*, Leuven, Belgium, May 1998.

[324] Ulrich, I., Nourbakhsh, I., “Appearance obstacle detection with monocular color vision,” *the Proceedings of the AAAI National Conference on Artificial Intelligence*. Austin, TX. August 2000.

[325] Ulrich, I., Nourbakhsh, I., “Appearance-based place recognition for topological localization,” *Proceedings of t he IEEE International Conference on Robotics and Automation*, San Francisco, 1023–1029, April 2000.

[326] Vanualailai, J., Nakagiri, S., Ha, J-H., “Collision avoidance in a two-point system via Liapunov’s second method,” *Mathematics and Simulation* 39: 125–141, 1995.

[327] Van Winnendael, M., Visenti G., Bertrand, R., Rieder, R., “Nanokhod microrover heading towards Mars,” *Proceedings of the Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space* (ESA SP-440), Noordwijk, Netherlands, 1999.

[328] Vandorpe, J., Brussel, H. V., Xu, H. “Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range finder,” *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA, 901–908, 1996.

[329] Weiss, G., Wetzler, C., Puttkamer, E., “Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’94)*, Munich, September 1994.

[330] Weingarten, J., Gruener, G. and Siegwart, R., “A state-of-the-art 3D sensor for robot navigation,” *Proceedings of IROS*, Sendai, September 2004.

[331] Weingarten, J. and Siegwart, R., “3D SLAM using planar segments,” Proceedings of IROS, Beijing, October 2006.

[332] Wullschleger, F.H., Arra,s K.O., Vestli, S.J., “A flexible exploration framework for map building,” *Proceedings of the Third European Workshop on Advanced Mobile Robots (Eurobot 99)*, Zurich, September 1999.

[333] Yagi, Y., Kawato, S.,“Panorama scene analysis with conic projection,” *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Workshop on Towards a New Frontier of Applications*, 1990.

[334] Yamauchi, B., Schultz, A., Adams, W., “Mobile robot exploration and map-building with continuous localization,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA’98)*, Leuven, Belgium, May 1998.

[335] Ying, X., Hu, Z., "Can we consider central catadioptric cameras and fisheye cameras within a unified imaging model?," *European Conference on Computer Vision* (ECCV), Lecture Notes in Computer Science, Springer Verlag, May 2004.

[336] Zhang, L., Ghosh, B. K., "Line segment based map building and localization using 2D laser rangefinder," *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[337] Zhang, Z., "A flexible new technique for camera calibration," *Microsoft Research Technical Report 98-71*, December 1998
see also http://research.microsoft.com/~zhang.

Referenced Webpages

[338] Fisher, R.B. (editor), "CVonline: On-line Compendium of Computer Vision," Available at www.dai.ed.ac.uk/CVonline.

[339] The Intel Image Processing Library/Integrated Performance Primitives (Intel IPP): http://software.intel.com/en-us/intel-ipp.

[340] Source code release site: www.cs.cmu.edu/~jbruce/cmvision.

[341] Newton Labs website: www.newtonlabs.com.

[342] For probotics: http://www.personalrobots.com.

[343] OpenCV, the Open Source Computer Vision library: http://opencv.willowgarage.com/wiki.

[344] Passive walking: www-personal.umich.edu/~artkuo/Passive_Walk/passive_walking.html.

[345] Passive walking, the Cornell Ranger: http://ruina.tam.cornell.edu/research/topics/locomotion_and_robotics/ranger/ranger2008.php.

[346] Computer Vision industry: http://www.cs.ubc.ca/spider/lowe/vision.html.

[347] Camera Calibration Toolbox for Matlab: http://www.vision.caltech.edu/bouguetj/calib_doc.

[348] List of camera calibration softwares: http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/links.html.

[349] Omnidirectional camera calibration toolbox from Christopher Mei http://www.robots.ox.ac.uk/~cmei/Toolbox.html.

[350] Omnidirectional camera calibration toolbox from Joao Barreto http://www.isr.uc.pt/~jpbar/CatPack/pag1.htm.

[351] Omnidirectional camera calibration toolbox from Davide Scaramuzza:
google "ocamcalib" or go to http://robotics.ethz.ch/~scaramuzza/Davide_Scaramuzza_files/Research/OcamCalib_Tutorial.htm.

[352] Open source software for SLAM and loop-closing: http://openslam.org.

[353] Open source software for multi-view structure from motion: http://phototour.cs.washington.edu/bundler

[354] Microsoft Photosynth: http://photosynth.net

[355] Photo Tourism: http://phototour.cs.washington.edu/

[356] Voodoo Camera Tracker: A tool for the integration of virtual and real scenes http://www.digilab.uni-hannover.de/docs/manual.html

[357] Augmented-reality toolkit (ARToolkit): http://www.hitl.washington.edu/artoolkit
[358] Parallel Tracking and Mapping (PTAM): http://www.robots.ox.ac.uk/~gk/PTAM