22. Range Sensors

Robert B. Fisher, Kurt Konolige

Range sensors are devices that capture the three-dimensional (3-D) structure of the world from the viewpoint of the sensor, usually measuring the depth to the nearest surfaces. These measurements could be at a single point, across a scanning plane, or a full image with depth measurements at every point. The benefits of this range data is that a robot can be reasonably certain where the real world is, relative to the sensor, thus allowing the robot to more reliably find navigable routes, avoid obstacles, grasp objects, act on industrial parts, etc.

This chapter introduces the main representations for range data (point sets, triangulated surfaces, voxels), the main methods for extracting usable features from the range data (planes, lines, triangulated surfaces), the main sensors for acquiring it (Sect. 22.1 – stereo and laser triangulation and ranging systems), how multiple observations of the scene, e.g., as if from a moving robot, can be registered (Sect. 22.2), and several indoor and outdoor robot applications where range data greatly simplifies the task (Sect. 22.3).

22.1 Range Sensing Basics

Here we present: (1) the basic representations used for range image data, (2) a brief introduction to the main 3-D sensors that are less commonly used in robotics applications, and (3) a detailed presentation of the commoner laser-based range image sensors.

22.1.1 Range Images and Point Sets

Range data is a $2\frac{1}{2}$-D or 3-D representation of the scene around the robot. The 3-D aspect arises because we are measuring the (X, Y, Z) coordinates of one or more points in the scene. Often only a single range image is used at each time instance. This means that we only observe the front sides of objects – the portion of the scene visible from the robot. In other words, we do not have a full 3-D observation of all sides of a scene. This is the origin of the term $2\frac{1}{2}$-D. Figure 22.1a shows a sample range image and Fig. 22.1b shows a registered reflectance image, where each pixel records the level of reflected infrared light.

There are two standard formats for representing range data. The first is an image $d(i, j)$, which records the distance d to the corresponding scene point (X, Y, Z)

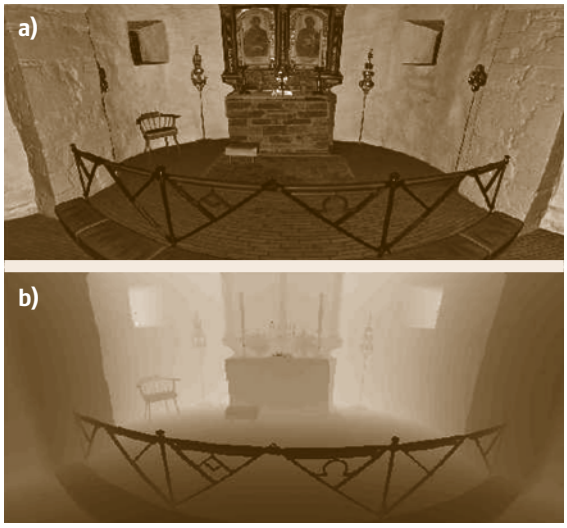

Fig. 22.1 **(a)** Registered infrared reflectance image. **(b)** Range image where closer is darker

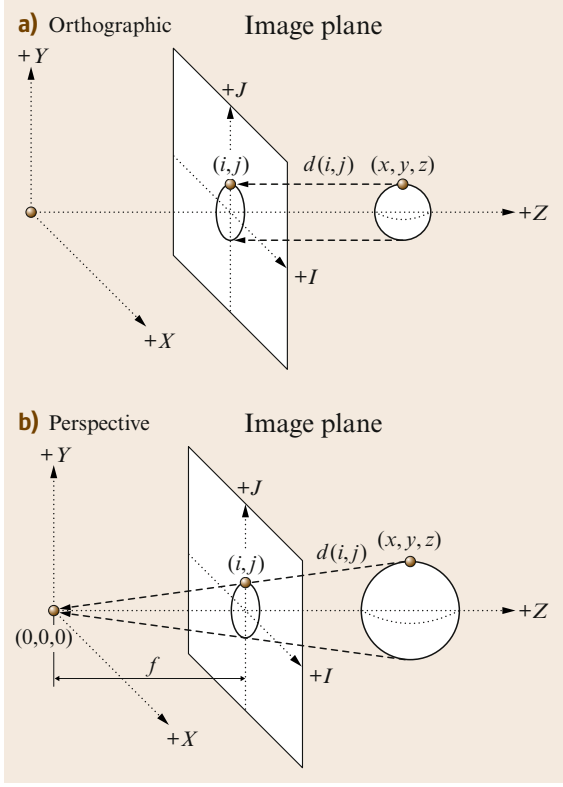

Fig. 22.2a,b Different range image mappings: **(a)** orthographic and **(b)** perspective

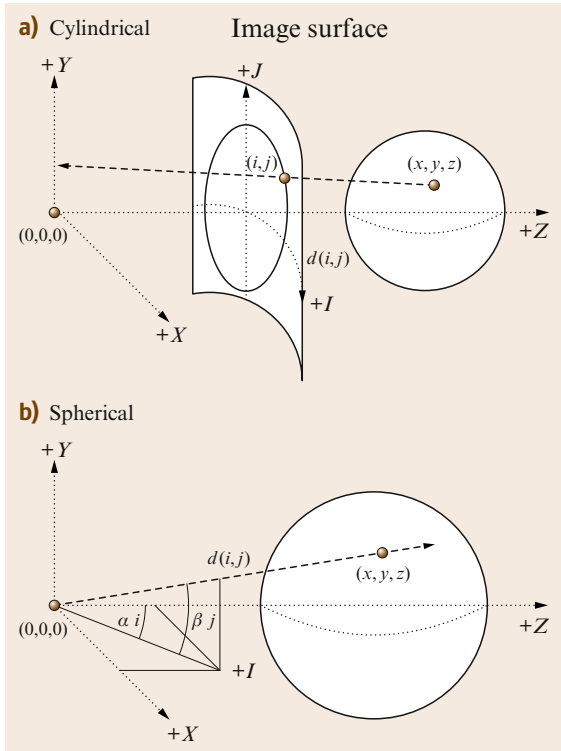

Fig. 22.3a,b Different range image mappings: **(a)** cylindrical and **(b)** spherical

for each image pixel (i, j). There are several common mappings from $(i, j, d(i, j))$ to (X, Y, Z), usually arising from the geometry of the range sensor. The most common image mappings are illustrated in Figs. 22.2 and 22.3. In the formulas given here, α and β are calibrated values specific to the sensor.

1. *Orthographic*. Here $(X, Y, Z) = (\alpha i, \beta j, d(i, j))$. These images often arise from range sensors that scan by translating in the x and y directions. (Fig. 22.2a.)
2. *Perspective*. Here $d(i, j)$ is the distance along the line of sight of the ray through pixel (i, j) to point (x, y, z). Treating the range sensor focus as the origin $(0, 0, 0)$ and assuming that its optical axis is the Z axis and that the (X, Y) axes are parallel to the image (i, j) axes, then $(X, Y, Z) = d(i, j)/\sqrt{\alpha^2 i^2 + \beta^2 j^2 + f^2}(\alpha i, \beta j, f)$, where f is the *focal length* of the system. These images often arise from sensor equipment that incorporates a normal intensity camera. (Fig. 22.2b.)

3. *Cylindrical*. Here, $d(i, j)$ is the distance along the line of sight of the ray through pixel (i, j) to point (X, Y, Z). In this case, the sensor usually rotates to scan in the x direction, and translates to scan in the y direction. Thus, $(X, Y, Z) = (d(i, j)\sin(\alpha i), \beta j, d(i, j)\cos(\alpha i))$ is the usual conversion. (Fig. 22.3a.)
4. *Spherical*. Here, $d(i, j)$ is the distance along the line of sight of the ray through pixel (i, j) to point (X, Y, Z). In this case, the sensor usually rotates to scan in the x direction, and, once each x scan, also rotates in the y direction. Thus (i, j) are the azimuth and elevation of the line of sight. Here $(X, Y, Z) = d(i, j)(\cos(\beta j)\sin(\alpha i), \sin(\beta j), \cos(\beta j)\cos(\alpha i))$. (Fig. 22.3b.)

Some sensors only record distances in a plane, so the scene (x, z) is represented by the linear image $d(i)$ for each pixel i. The orthographic, perspective, and cylindrical projection options listed above still apply in simplified form.

The second format is as a list $\{(X_i, Y_i, Z_i)\}$ of 3-D data points, but this format can be used with all of the mappings listed above. Given the conversions from image data $d(i, j)$ to (X, Y, Z) the range data is only supplied as a list. Details of the precise mapping and data format are supplied with commercial range sensors.

22.1.2 Stereo Vision

It is possible to acquire range information from many different sensors, but only a few have the reliability needed for most robotics applications. The more reliable ones, laser-based triangulation and laser radar (LIDAR), are discussed in the next section.

Real-time stereo analysis uses two or more input images to estimate the distance to points in a scene. The basic concept is *triangulation*: a scene point and the two camera points form a triangle, and knowing the baseline between the two cameras, and the angle formed by the camera rays, the distance to the object can be determined.

In practice, there are many difficulties in making a stereo imaging system that is useful for robotics applications. Most of these difficulties arise in finding reliable matches for pixels in the two images that correspond to the same point in the scene. A further consideration is that stereo analysis for robotics has a real-time constraint, and the processing power needed for some algorithms can be very high. However, in recent years much progress has been made, and the advantage of stereo imaging is that it can provide full 3-D range images, registered with visual information, potentially out to an infinite distance, at high frame rates – something which no other range sensor can match.

In this subsection we will review the basic algorithms of stereo analysis, and highlight the problems and potential of the method. For simplicity, we use binocular stereo.

Stereo Image Geometry

This subsection gives some more detail of the fundamental geometry of stereo, and in particular the relationship of the images to the 3-D world via projection and reprojection. A more in-depth discussion of the geometry, and the rectification process, can be found in [22.2].

The input images are *rectified*, which means that the original images are modified to correspond to ideal pinhole cameras with a particular geometry, illustrated in Fig. 22.4. Any 3-D point S projects to a point in the images along a ray through the focal point. If the principal rays of the cameras are parallel, and the images are

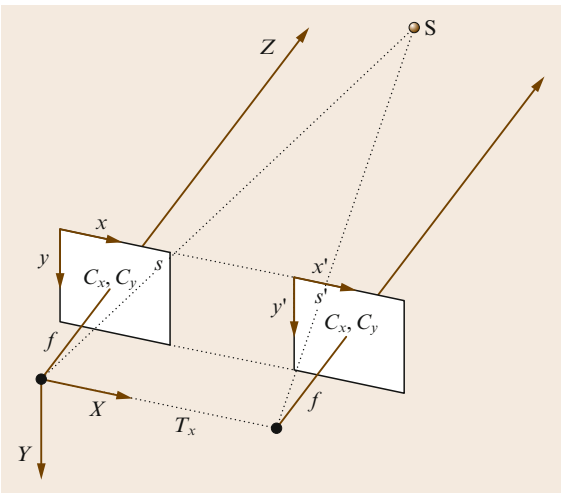

Fig. 22.4 Ideal stereo geometry. The global coordinate system is centered on the focal point (camera center) of the left camera. It is a right-handed system, with positive Z in front of the camera, and positive X to the right. The camera principal ray pierces the image plane at C_x, C_y, which is the same in both cameras (a variation for *verged* cameras allows C_x to differ between the images). The focal length is also the same. The images are lined up, with $y = y'$ for the coordinates of any scene point projected into the images. The difference between the x coordinates is called the *disparity*. The vector between the focal points is aligned with the X-axis [22.1]

embedded in a common plane and have collinear scan lines, then the search geometry takes a simple form. The *epipolar line* of a point s in the left image, defined as the possible positions of s' in the right image, is always a scan line with the same y coordinate as s. Thus, the search for a stereo match is linear. The process of finding a rectification of the original images that puts them into standard form is called *calibration*, and is discussed in [22.2].

The difference in the x coordinates of s and s' is the *disparity* of the 3-D point, which is related to its distance from the focal point, and the baseline T_x that separates the focal points.

A 3-D point can be projected into either the left or right image by a matrix multiplication in homogenous coordinates, using the *projection matrix*. The 3-D coordinates are in the frame of the left camera (Fig. 22.4).

$$\boldsymbol{P} = \begin{pmatrix} F_x & 0 & C_x & -F_x T_x \\ 0 & F_y & C_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} . \tag{22.1}$$

This is the projection matrix for a single camera. F_x and F_y are the focal lengths of the rectified images, and C_x, C_y is the optical center. T_x is the translation of the camera relative to the left (reference) camera. For the left camera, it is 0; for the right camera, it is the baseline times the x focal length.

A point in three dimensions is represented by homogeneous coordinates and the projection is performed using a matrix multiplication

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \boldsymbol{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} , \tag{22.2}$$

where $(x/w, y/w)$ are the idealized image coordinates.

If points in the left and right images correspond to the same scene feature, the depth of the feature can be calculated from the image coordinates using the *reprojection matrix*

$$\boldsymbol{Q} = \begin{pmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & F_x \\ 0 & 0 & -1/T_x & (C_x - C_{x'})/T_x \end{pmatrix} . \tag{22.3}$$

The primed parameters are from the left projection matrix, the unprimed from the right. The last term is zero except for verged cameras. If x, y and x', y are the two matched image points, with $d = x - x'$, then

$$\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \boldsymbol{Q} \begin{pmatrix} x \\ y \\ d \\ 1 \end{pmatrix} , \tag{22.4}$$

where $(X/W, Y/W, Z/W)$ are the coordinates of the scene feature, and $d = x' - x$ is the disparity. Assuming $C_x = C'_x$, the Z distance assumes the familiar inverse form of triangulation

$$Z = \frac{F - xT_x}{d} . \tag{22.5}$$

Reprojection is valid only for rectified images – for the general case, the projected lines do not intersect. The disparity d is an *inverse depth* measure, and the vector (x, y, d) is a perspective representation range image (see Sect. 22.1.1), sometimes called the *disparity space* representation. The disparity space is often used in applications instead of 3-D space, as a more efficient representation for determining obstacles or other features (see Sect. 22.3.3).

Equation (22.4) is a homography between disparity space and 3-D Euclidean space. Disparity space is also useful in translating between 3-D frames. Let $p_0 = (x_0, y_0, d_0, 1)$ in frame 0, with frame 1 related by the rigid motion R, t. From the reprojection (22.4) the 3-D position is $\boldsymbol{Q}\, p_0$. Under the rigid motion this becomes $\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \boldsymbol{Q}\, p_0$, and finally applying $\boldsymbol{Q}^{-1}$ yields the disparity representation in frame 1. The concatenation of these operations is the homography

$$H(R, t) = \boldsymbol{Q}^{-1} \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \boldsymbol{Q} . \tag{22.6}$$

Using the homography allows the points in the reference frame to be directly projected onto another frame, without translating to 3-D points.

Stereo Methods

The fundamental problem in stereo analysis is matching image elements that represent the same object or object part in the scene. Once the match is made, the range to the object can be computed using the image geometry.

Matching methods can be characterized as local or global. Local methods attempt to match small regions of one image to another based on intrinsic features of the region. Global methods supplement local methods by considering physical constraints such as surface continuity or base of support. Local methods can be further

classified by whether they match discrete features among images, or correlate a small area patch [22.3]. Features are usually chosen to be lighting and viewpoint independent, e.g., corners are a natural feature to use because they remain corners in almost all projections. Feature-based algorithms compensate for viewpoint changes and camera differences, and can produce rapid, robust matching, but they have the disadvantage of requiring perhaps expensive feature extraction, and yielding only sparse range results.

In the next section we present local area correlation in more detail, since it is one of the most efficient and practical algorithms for real-time stereo. A survey and results of recent stereo matching methods is in [22.5], and the authors maintain a web page listing up-to-date information [22.6].

Area Correlation Stereo

Area correlation compares small patches among images using correlation. The area size is a compromise, since small areas are more likely to be similar in images with different viewpoints, while larger areas increase the signal-to-noise ratio. In contrast to the feature-based method, area-based correlation produces dense results. Because area methods need not compute features, and have an extremely regular algorithmic structure, they can have optimized implementations.

The typical area correlation method has five steps (Fig. 22.5).

1. Geometry correction. In this step, distortions in the input images are corrected by warping into a *standard form.*
2. Image transform. A local operator transforms each pixel in the grayscale image into a more appropriate form, e.g., normalizes it based on average local intensity.
3. Area correlation. This is the correlation step, where each small area is compared with other areas in its search window.
4. Extrema extraction. The extreme value of the correlation at each pixel is determined, yielding a disparity image: each pixel value is the disparity between left and right image patches at the best match.
5. Postfiltering. One or more filters clean up noise in the disparity image result.

Correlation of image areas is disturbed by illumination, perspective, and imaging differences among images. Area correlation methods usually attempt to compensate by correlating not the raw intensity images, but some transform of the intensities. Let u, v be

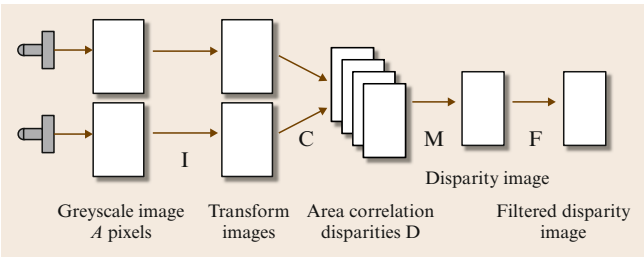

Fig. 22.5 Basic stereo processing. See text for details [22.4]

the center pixel of the correlation, x, y be pixels in the neighborhood about u, v the disparity, and $I_{x,y}$, $I'_{x,y}$ the intensities of the left and right images.

1. Normalized cross-correlation.
$$\frac{\sum_{x,y}[I_{x,y} - \hat{I}_{x,y}][I'_{x-d,y} - \hat{I'}_{x-d,y}]}{\sqrt{\sum_{x,y}[I_{x,y} - \hat{I}_{x,y}]^2 \sum_{x,y}[I'_{x,y} - \hat{I'}_{x-d,y}]^2}}$$
2. High-pass filter such as Laplacian of Gaussian (LOG). The Laplacian measures directed edge intensities over some area smoothed by the Gaussian. Typically the standard deviation of the Gaussian is 1–2 pixels.
$$\sum_{x,y} s(\log_{x,y} - \log_{x-d,y}) .$$
Where $s(x)$ is x^2 or $||x||$ and $\log_{x,y}$ is the Laplacian of Gaussian at pixl x, y.
3. Nonparametric. These transforms are an attempt to deal with the problem of outliers, which tend to overwhelm the correlation measure, especially using a square difference. The census method [22.7] computes a bit vector describing the local environment of a pixel, and the correlation measure is the Hamming distance between two vectors
$$\sum_{x,y} (I_{x,y} > I_{u,v}) \oplus (I'_{x-d,y} > I'_{u,v}) .$$

Results on the different transforms and their error rates for some standard images are compiled in [22.6].

Another technique for increasing the signal-to-noise ratio of matches is to use more than two images [22.8]. This technique can also overcome the problem of viewpoint occlusion, where the matching part of an object does not appear in the other image. The simple technique of adding the correlations between images at the same disparity seems to work well [22.9]. Obviously, the computational expenditure for multiple images is greater than for two.

Dense range images usually contain false matches that must be filtered, although this is less of a prob-

Table 22.1 Post-filtering techniques for eliminating false matches in area correlation

Correlation surface	[22.10]
Peak width	Wide peak indicates poor feature localization
Peak height	Small peak indicates poor match
Number of peaks	Multiple peaks indicate ambiguity
Mode filter	Lack of supporting disparities violates smoothness
Left/right check [22.11, 12]	Nonsymmetric match indicates occlusion
Texture [22.13]	Low texture energy yields poor matches

lem with multiple-image methods. Table 22.1 lists some of the postfilters that have been discussed in the literature.

Disparity images can be processed to give subpixel accuracy, by trying to locate the correlation peak between pixels. This increases the available range resolution without much additional work. Typical accuracies are a tenth of a pixel.

Stereo Range Quality

Various artifacts and problems affect stereo range images.

Smearing. Area correlation introduces expansion in foreground objects, e.g., the woman's head in Fig. 22.6. The cause is the dominance of strong edges on the object. Nonparametric measures are less subject to this phenomenon. Other approaches include multiple correlation windows and shaped windows.

Dropouts. These are areas where no good matches can be found because of low texture energy. Dropouts are a problem for indoor and outdoor manmade surfaces. Projecting an infrared random texture can help [22.14].

Range resolution. Unlike laser radar (LADAR) devices, stereo range accuracy is a quadratic function of distance, found by differentiating (22.5) with respect to disparity:

$$\delta Z = -\frac{F_x T_x}{d^2} \,. \tag{22.7}$$

The degradation of stereo range with distance can be clearly seen in the 3-D reconstruction of Fig. 22.6.

Processing. Area correlation is processor intensive, requiring Awd operations, where A is the image area, w the correlation window size, and d the number of disparities. Clever optimizations take advantage of redundant calculations to reduce this to Ad (independent of window size), at the expense of some storage. Real-time implementations exist for standard personal computers (PCs) [22.1, 15], graphics accelerators [22.16, 17], digital signal processors (DSPs) [22.4], field-programmable gate arrays (FPGAs) [22.1, 18], and specialized application-specific integrated circuits (ASICs) [22.19].

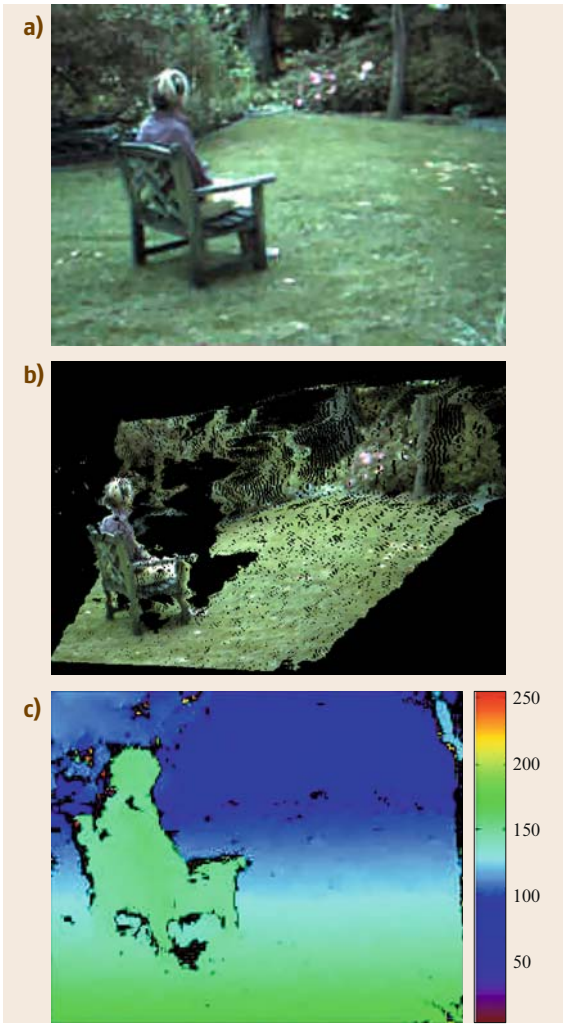

Fig. 22.6a–c Sample stereo results from an outdoor garden scene; baseline is 9 cm. **(a)** Original left image. **(b)** Computed 3-D points from a different angle. **(c)** Disparity in pseudo-color [22.1]

Other Visual Sources of Range Information

Here we briefly list the most popular, but less reliable sources of range information. These sources can potentially supplement other sensors.

- *Focus/defocus*. Knowledge of the camera parameters and the amount of blur of image features allows estimation of how far the corresponding scene features are from the perfect focus distance [22.20]. Sensors may be passive (using a precaptured image) or active (capturing several images with different focus settings).
- *Structure and motion*. Structure and motion algorithms compute 3-D scene structure and the sensor positions simultaneously [22.21]. These is essentially a binocular stereo process (see discussion before), except that only a single moving camera is used. Thus the images needed by the stereo process are acquired by the same camera in several different positions. Video camcorders are also used, but have lower resolution. One important advantage of this approach over the normal algorithm is that features can be tracked easily if the time between frames or the motion is small enough. This simplifies the *correspondence problem*, however it can lead to another problem. If the pair of images used for the stereo calculation are taken close together in time, then the separation between the cameras images will not be much, that is, there will be a *short baseline*. Triangulation calculations are then more inaccurate, as small errors in estimating the position of image features results in large errors in the estimated 3-D position (particularly the depth estimate). This problem can be partly avoided by tracking for longer periods. A second problem that can arise is that not all motions are suitable for estimating the full 3-D scene structure, for example, if the video recorder only rotates about its optical axis or about its focus point, then no 3-D information can be recovered. A little care can avoid this problem.
- *Shading*. The pattern of shading on a surface is related to the orientation of the surface relative to the observer and light sources. This relationship can be used to estimate the surface orientation across the surface. The surface normals can then be integrated to give an estimate of the relative surface depth.
- *Photometric stereo*. Photometric stereo [22.22] is a combination of shading and stereo processes. The key concept is that the shading of an object varies with the position of the light sources. Hence, if you had several aligned pictures of an object or scene with the light source in different positions (e.g., the sun moved), then you can calculate the scene's surface normals. From these, the relative surface depth can be estimated. The restriction of a stationary observer and changing light sources makes this approach less likely to be useful to most robotics applications.
- *Texture*. The way uniform or statistical textures vary on a surface is related to the orientation of the surface relative to the observer. As with shading, the texture gradients can be used to estimate the surface orientation across the surface [22.23]. The surface normals can then be integrated to give an estimate of the relative surface depth.

22.1.3 Laser-Based Range Sensors

There are three types of laser-based range sensors in common use:

1. triangulation sensors
2. phase-modulation sensors
3. time-of-flight sensors

These are discussed in more detail below. There are also Doppler and interference laser-based range sensors, but these seem to not be in general use at the moment so we skip them here. An excellent recent review of range sensors is by *Blais* [22.24].

The physical principles behind the three types of sensors discussed below do not intrinsically depend on the use of a laser – for the most part any light source would work. However, lasers are traditional because: (1) they can easily generate bright beams with lightweight sources, (2) infrared beams can be used unobtrusively, (3) they focus well to give narrow beams, (4) single-frequency sources allow easier rejection filtering of unwanted frequencies, (5) single-frequency sources do not disperse from refraction as much as full-spectrum sources, (6) semiconductor devices can more easily generate short pulses, etc.

One advantage of all three sensor types is that it is often possible to acquire a reflectance image registered with the range image. By measuring the amount that the laser beam strength has reduced after reflection from the target, one can estimate the reflectance of the surface. This is only the reflectance at the single spectral frequency of the laser, but, nonetheless, this gives useful information about the appearance of the surface (as well as the shape as given by the range measurement); Three-color laser systems [22.25] similarly give registered red–green–blue (RGB) color images. Figure 22.1 shows registered range and reflectance images from the same scene.

One disadvantage of all three sensor types is specular reflections. The normal assumption is that the observed light is a diffuse reflection from the surface. If the ob-

served surface is specular, such as polished metal or water, then the source illumination may be reflected in unpredictable directions. If any light eventually is detected by the receiver, then it is likely to cause incorrect range measurements. Specular reflections are also likely at the fold edges of surfaces.

A second problem is the laser *footprint*. Because the laser beam has a finite width, when it strikes the edge of a surface, part of the beam may be actually lie on a more distant surface. The consequences of this will depend on the actual sensor, but it commonly results in range measurements that lie between the closer and more distant surfaces. These measurements thus suggest that there is surface where there really is empty space. Some noise removal is possible in obvious empty space, but there may be erroneous measurements that are so close to the true surface that they are difficult to eliminate.

22.1.4 Time-of-Flight Range Sensors

Time-of-flight range sensors are exactly that: they compute distance by measuring the time that a pulse of light takes to travel from the source to the observed target and then to the detector (usually collocated with the source). In a sense, they are radar sensors that are based on light. The travel time multiplied by the speed of light (in the given medium – space, air or water – and adjusted for the density and temperature of the medium) gives the distance. Laser-based time-of-flight range sensors are also caller light detection and ranging (LIDAR) or laser radar (LADAR) sensors.

Limitations on the accuracy of these sensors are based on the minimum observation time (and thus the minimum distance observable), the temporal accuracy (or quantization) of the receiver, and the temporal width of the laser pulse.

Many time-of-flight sensors used for local measurements have what is called an *ambiguity interval*, for example, 20 m. The sensor emits pulses of light periodically, and computes an average target distance from the time of the returning pulses. To limit noise from reflections and simplify the detection electronics, many sensors only accept signals that arrive within time Δt, but this time window might also observe previous pulses reflected by more distant surfaces. This means that a measurement Z is ambiguous to the multiple of $1/2c\Delta t$ because surfaces that are further away (e.g., z) than $1/2c\Delta t$ are recorded as $z \bmod 1/2c\Delta t$. Thus distances on smooth surfaces can increase until they reach $c\Delta t/2$ and then they become 0. Typical values for $c\Delta t/2$ are 20–40 m. On smooth surfaces, such as a ground plane or road, an unwinding algorithm can recover the true depth by assuming that the distances should be changing smoothly.

Most time-of-flight sensors transmit only a single beam, thus range measurements are only obtained from a single surface point. Robotics applications usually need more information, so the range data is usually supplied as a vector of range to surfaces lying in a plane (Fig. 22.7) or as an image (Fig. 22.1). To obtain these denser representations, the laser beam is swept across the scene. Normally the beam is swept by a set of mirrors rather than moving the laser and detector themselves (mirrors are lighter and less prone to motion damage). The most common technologies for this are using a stepper motor (for program-based range sensing) or rotating or oscillating mirrors for automatic scanning.

Typical ground-based time-of-flight sensors suitable for robotics applications have a range of 10–100 m, and an accuracy of 5–10 mm. The amount of the scene scanned will depend on the sweep rate of the mirrors and the pulse rate, but 1–25 K points per second are typical. Manufacturers of these sensors include Acuity, SICK, Mensi, DeltaSphere, and Cyrax.

Recently, a type of time-of-flight range sensor called the *flash LADAR* has been developed. The key innovation has been the inclusion of timing circuits at each pixel of the sensor chip. Thus, each pixel can measure the time at which a light pulse is observed from the line of sight viewed by that pixel. This allows simultaneous calculation of the range values at each pixel. The light pulse now has to cover to whole portion of the scene that is observed, so sensors typically use an array of infrared laser light-emitting diodes (LEDs). While spatial resolution is smaller than current cameras (e.g., 64×64, 160×124, 128×128), the data can be acquired at video rates (30–50 fps), which provides considerable informa-

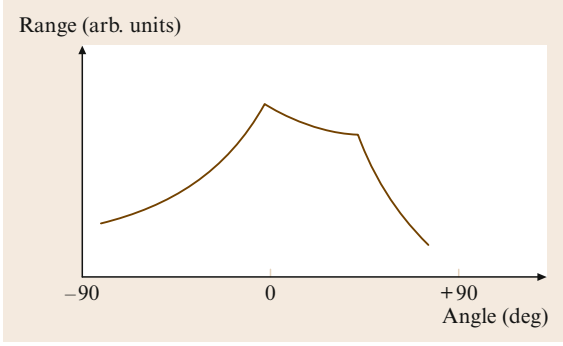

Fig. 22.7 Plot of ideal one-dimensional range image of sample distance versus angle of measurement

tion usable for robot feedback. Different sensor ranges have been reported, such as up to 5 m [22.26] (with on the order of 5–50 cm standard deviation depending on target distance and orientation) or at 1.1 km [22.27] (no standard deviation reported).

22.1.5 Modulation Range Sensors

Modulation-based range sensors are commonly of two types, where a continuous laser signal is either amplitude or frequency modulated. By observing the phase shift between the outgoing and return signals, the signal transit time is estimated and from this the target distance. As the signal phase repeats every 2π, these sensors also have an ambiguity interval.

These sensors also produce a single beam that must be swept. Scan ranges are typically 20–40 m and the accuracy is 5 mm. Figure 22.1 was captured with a modulation sensor.

22.1.6 Triangulation Range Sensors

Triangulation range sensors [22.28] are based on principles similar to the stereo sensors discussed previously. The key concept is illustrated in Fig. 22.8: a laser beam is projected from one position onto the observed surface. The light spot that this creates is observed by a sensor from a second position. Knowing the relative positions and orientations of the laser and sensor plus some simple trigonometry allows calculation of the 3-D position of the illuminated surface point. The triangulation process is more accurate when the laser spot's position is accurately estimated. About 0.1 pixel accuracy can be normally achieved [22.29].

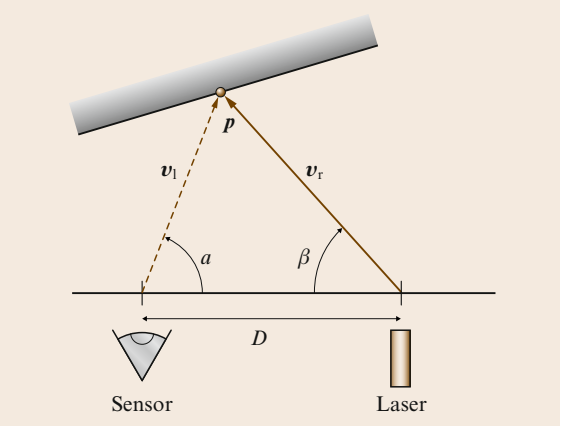

Fig. 22.8 Triangulation using a laser spot

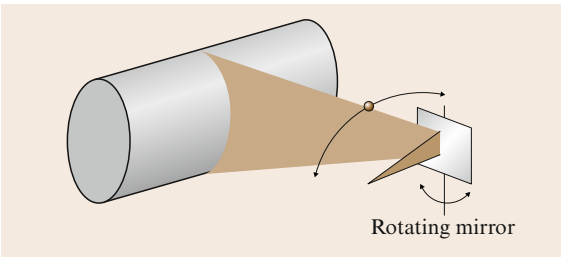

Fig. 22.9 A swept laser plane covers a larger scene

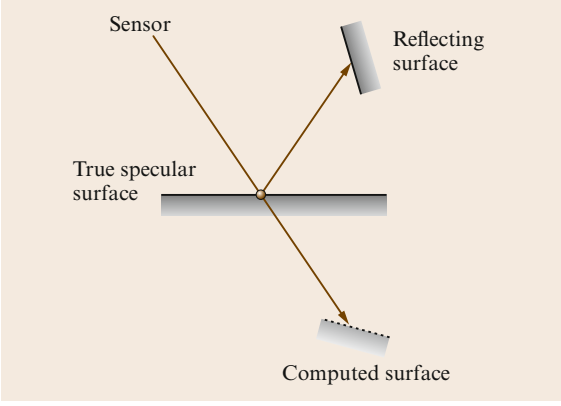

Fig. 22.10 Incorrect estimated depths on specular surfaces

Because the laser illumination can be controlled, this gives a number of practical advantages.

1. A laser of known frequency (e.g., 733 nm) can be matched with a very selective spectral filter of the same frequency (e.g., 5 nm half-width). This normally allows unique identification of the light spot as the filtering virtually eliminates all other bright light, leaving the laser spot as the brightest spot in the image.
2. The laser spot can be reshaped with lenses or mirrors to create multiple spots or stripes, thus allowing the measurement of multiple 3-D points simultaneously. Stripes are commonly used because these can be swept across the scene (see Fig. 22.9 for an example) to observe a full scene. Other illumination patterns are also commonly used, such as parallel lines, concentric circles, cross hairs, and dot grids. Commercial structured light pattern generators are available from, e.g., Lasiris or Edmunds Optics.
3. The laser light ray can be positioned by mirrors under computer control to selectively observe a given area, such as around a doorway or potential obstacle, or an object that might be grasped.

Disadvantages include

1. potential eye safety risks from the power of lasers, particularly when invisible laser frequencies are used (commonly infrared),
2. specular reflections from metallic and polished objects, which may cause incorrect calculation of where the laser spot is illuminating, as in Fig. 22.10, where the hypothesized surface lies behind the true surface.

22.1.7 Example Sensors

A typical phase-modulation-based range sensor is the Zohler and Frohlich 25200. This is an expensive spherical scan sensor, observing a full 360° horizontally and 155° vertically. Each scan can acquire up to 20 000 3-D points horizontally and 8000 points vertically in about 100 s. The accuracy of the scan depends on the distance to the target, but 4 mm is typical, with samples every 0.01° (samples every 1.7 mm at 10 m distance). The densely sensed data is typically used for 3-D scene survey, modeling, and virtual-reality reconstruction.

An example of a laser sensor that has been commonly used for robot navigation is the SICK LMS 200. This sensor sweeps a laser ray in a horizontal plane through an arc of 180°, acquiring 720 range measurements in a plane in 0.05 s. While a plane might seem limiting, the observed 3-D data can be easily matched to a stored (or previously learned) model of the environment at the height of the scan. This allows accurate estimation of the sensor's position in the scene (and thus a mobile vehicle that carries the sensor). Of course, this requires an environment without overhanging structures that are not detectable at the scan height. Another common configuration is to mount the sensor high on the vehicle with the scan plane tilted downwards. In this configuration the ground in front of the vehicle is progressively scanned as the vehicle moves forward, thus allowing detection of potential obstacles.

The Minolta 910 is a small triangulation scanner with an accuracy of about 0.01 mm in a range of up to 2 m, for about 500^2 points in 2.5 s. It has commonly been used for small-region scanning, such as for inspection or part modeling, but can also be mounted in a fixed location to scan a robot workcell. Observation of the workcell allows a variety of actions, such as inspection of parts, location of dropped or bin delivered components, or accurate part position determination.

More examples and details of range sensors for robot navigation are presented in Sect. 22.3.

22.2 Registration

This section introduces techniques for 3-D localization of parts for robot manipulation, self-localization of robot vehicles, and scene understanding for robot navigation. All of these are based on the ability to register 3-D shapes, e.g., range images to range images, triangulated surfaces or geometric models. Before we introduce these applications in Sect. 22.2.7, we first look at some basic techniques for manipulating 3-D data.

22.2.1 3-D Feature Representations

There are many representations available for encoding 3-D scene structure and model representations, but the following representations are the ones most commonly encountered in robotics applications. Some scene models or descriptions may use more than one of these simultaneously to describe different aspects of the scene or object models.

3-D Point Sets

This is a set $\{\boldsymbol{p}_i = (X_i, Y_i, Z_i)\}$ of 3-D points that describe some salient and identifiable points in the scene. They might be the centers of spheres (often used as markers), corners where three planes intersect, or the extrema of some protrusion or indentation on a surface. They may be a subset of an initially acquired 3-D full scene point set, they might be extracted from a range image, or they might be computed theoretical points based on extracted data features.

Planes

This is a set of planar surfaces observed in a scene or bounding part or all of an object model. There are several ways to represent a plane, but one common representation is by the equation of the plane $ax + by + cz + d = 0$ that passes through corresponding scene or model surface points $\{\boldsymbol{p}_i = (X_i, Y_i, Z_i)\}$. A plane only needs three parameters, so there are usually

some additional constraints on the plane representation (a, b, c, d), such as $d = 1$ or $a^2 + b^2 + c^2 = 1$. The vector $\boldsymbol{n} = (a, b, c)'/a^2 + b^2 + c^2$ is the surface normal.

A planar surface may only be described by the infinite surface as given by the equation, but it may also include a description of the boundary of the surface patch. Again there are many possible representations for curves, either in three or two dimensions lying in the plane of the surface. Convenient representations for robotics applications are lists of the 3-D points $\{(X_i, Y_i, Z_i)\}$ that form the patch boundary, or polylines, which represent the boundary by a set of connected line segments. A polyline can be represented by the sequence of 3-D points $\{(X_i, Y_i, Z_i)\}$ that form the vertices that join the line segments.

Triangulated Surfaces

This representation describes an object or scene by a set of triangular patches. More general polygonal surface patches or even various smooth surface representations are also used, but triangles are most commonly used because they are simpler and there are inexpensive PC graphics cards that display triangles at high speed.

The triangles can be large (e.g., when representing planar surfaces) or small (e.g., when representing curved surfaces). The size chosen for the triangles reflects the accuracy desired for representing the object or scene surfaces. The triangulated surface might be complete in the sense that all observable scene or objects surfaces are represented by triangles, or there might be disconnected surface patches with or without internal holes. For grasping or navigation, you do not want any unrepresented scene surface to lie in front of the represented portion of the surface where a gripper or vehicle might collide with it. Hence, we assume that the triangulation algorithms produce patch sets that, if completely connected at the edges, implicitly bound all real scene surfaces. Figure 22.11 shows an example of a triangulated surface over the original underlying smooth surface.

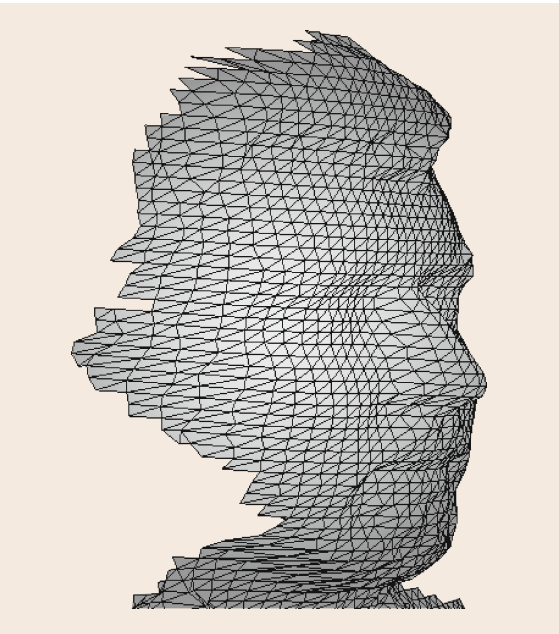

Fig. 22.11 Triangulated surface (thanks to T. Breckon)

There are many slightly different schemes for representing the triangles [22.30], but the main features are a list $\{\boldsymbol{v}_i = (X_i, Y_i, Z_i)\}$ indexed by i of the 3-D vertices at the corners of the triangles, and a list of the triangles $\{\boldsymbol{t}_n = (i_n, j_n, k_n)\}$ specifying the indices of the corner vertices.

From the triangular surface patches, one can compute potential grasping positions, potential navigable surfaces or obstacles, or match observed surface shapes to previously stored shapes.

3-D Lines

The 3-D lines where planar surfaces meet are features that can be easily detected by both stereo and range sensors. These features occur commonly in built environments (e.g., where walls, floors, ceilings, and doorways meet, around the edges of wall structures like notice boards, at the edges of office and warehouse furniture, etc.). They are also common on manmade objects. In the case of stereo imaging, changes of surface shape or coloring are detected as edges, which can be matched in the stereo process to directly produce the 3-D edge. In the case of a range sensor, planar surfaces can be easily extracted from the range data (see next section) and adjacent planar surfaces can be intersected to give the edges.

The most straightforward representation for 3-D lines is the set of points $\boldsymbol{x} = \boldsymbol{p} + \lambda \boldsymbol{v}$ for all λ, where $\boldsymbol{v}$ is a unit vector. This has five degrees of freedom; more complex representations, e.g., with four degrees of freedom exist [22.2].

Voxels

The voxel (volume pixel) approach represents the 3-D world by 3-D boxes/cells that indicate where there is scene structure and where there is free space. The simplest representation is a 3-D binary array, encoded as 1 for having structure and 0 for free space. This can be quite memory intensive, and also requires a lot of

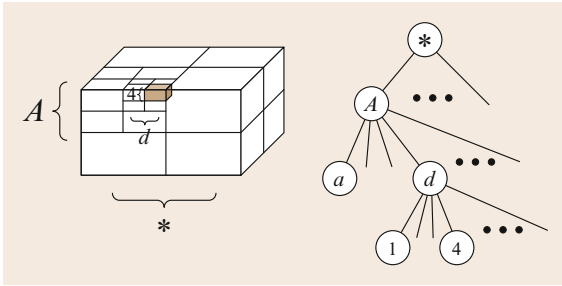

Fig. 22.12 Recursive space decomposition and a portion of the corresponding octree

computation to check many voxels for content. A more complex but more compact representation is the hierarchical representation called the octree [22.30]. This divides the entire (bounded) rectangular space into eight rectangular subspaces called octants (Fig. 22.12). A tree data structure encodes the content of each octant as empty, full or mixed. Mixed octants are then subdivided into eight smaller rectangular octants, encoded as subtrees of the larger tree. Subdivision continues until some minimum octant size is reached. Determining whether a voxel is empty, full or mixed depends on the sensor used, however, if no 3-D data points are located in the volume of a voxel, then it is likely to be empty. Similarly, if many 3-D points are present, then the voxel is likely to be full. The remaining voxels can be considered to be mixed. More details of a voxelization algorithm can be found in [22.31].

For the purpose of robot navigation, localization or grasping, only the surface and free-space voxels need to be marked accurately. The interior of objects and scene structure are largely irrelevant.

22.2.2 3-D Feature Extraction

Three types of structured 3-D features are of particular interest for robot applications: straight lines, planes, and triangulated surfaces. How these are extracted from range data sets is summarized here:

Planes

Planes are generally found by some region-growing process, based on a seed patch. When the scene largely consists of planes, a particularly simple approach is based on selecting a previously unused point and the set of points $\{\boldsymbol{v}_i = (X_i, Y_i, Z_i)\}$ in its neighborhood. A plane is fit to these points using a least-squares method: create a $4 \times N$ matrix $\boldsymbol{D}$ of the N points in the set, each point augmented with a 1. Call these extended points $\{\boldsymbol{v}_i\}$. As each point lies in the same plane $ax + by + cz + d = 0$, we can easily estimate the plane parameter vector $\boldsymbol{p} = (a, b, c, d)'$ as follows. We know that a perfect plane fit to perfect data would satisfy $\boldsymbol{Dp} = \boldsymbol{0}$. Since there is noise, we instead create the error vector $\boldsymbol{e} = \boldsymbol{Dp}$ and find the vector $\boldsymbol{p}$ that minimizes its length (squared) $\boldsymbol{e} \cdot \boldsymbol{e} = \boldsymbol{p}'\boldsymbol{D}'\boldsymbol{Dp}$. This vector is the eigenvector with the smallest eigenvalue of the matrix $\boldsymbol{D}'\boldsymbol{D}$. This initial hypothesized plane then needs to be tested for reasonableness by (1) examining the smallest eigenvalue, which should be small and on the order of the square of the expected noise level, and (2) ensuring that most of the 3-D points in the fitted set lie on the plane (i. e., $| \boldsymbol{v}_i \cdot \boldsymbol{p} | < \tau$).

Larger planar regions are *grown* by locating new adjacent points $\boldsymbol{v}_i$ that lie on the plane (i. e., $| \boldsymbol{v}_i \cdot \boldsymbol{p} | < \tau$). When enough of these are found, the parameters $\boldsymbol{p}$ of the plane are re-estimated. Points on the detected plane are removed and the process is repeated with a new seed patch. This process continues until no more points can be added. More complete descriptions of planar feature extraction can be found in [22.32].

When the planes are more sparsely found in the scene, then another approach is the random sample consensus (RANSAC) feature detection algorithm [22.33]. When adapted for plane finding, this algorithm selects three 3-D points at random (although exploiting some locality to the point selection algorithm can improve the efficiency of the algorithm). These three points determine a plane with parameter vector $\boldsymbol{p}$. All points $\{\boldsymbol{v}\}$ in the set are testing for belonging to the plane (i. e., $| \boldsymbol{v}_i \cdot \boldsymbol{p} | < \tau$). If enough points are close to the plane, then potentially a plane has been found. These points should also be processed to find a connected set, from which a more accurate set of plane parameters can be estimated using the least-squares algorithm given above. If a planar patch is successfully found, the points that lie in that plane are removed from the dataset. The random selection of three points then continues until no more planes are found (a bound on how many tries to make can be estimated).

Straight Lines

While straight lines are common in manmade scenes, direct extraction from 3-D datasets is not easy. The main source of the difficulty is that 3-D sensors often do not acquire good responses at edges of surfaces as (1) the sensor beam will be sampling from two different surfaces and 2) laser-based sensors produce somewhat unpredictable effects. For this reason, most 3-D line detection algorithms are indirect, whereby planes are first

detected, e.g., using the method of the previous section, and then adjacent planes are intersected. Adjacency can be tested by finding paths of connected pixels that lead from one plane to the other. If planes 1 and 2 contains points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ and have surface normals $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$, respectively, then the resulting intersection line has equation $\boldsymbol{x} = \boldsymbol{a} + \lambda \boldsymbol{d}$, where $\boldsymbol{a}$ is a point on the line and $\boldsymbol{d} = \boldsymbol{n}_1 \times \boldsymbol{n}_2 / \,||\, \boldsymbol{n}_1 \times \boldsymbol{n}_2 \,||$ is the line direction. There are an infinite number of possible points $\boldsymbol{a}$, which can be found by solving the equations $\boldsymbol{a}'\boldsymbol{n}_1 = \boldsymbol{p}_1'\boldsymbol{n}_1$ and $\boldsymbol{a}'\boldsymbol{n}_2 = \boldsymbol{p}_2'\boldsymbol{n}_2$. A reasonable third constraint that obtains a point near to $\boldsymbol{p}_2$ is the equation $\boldsymbol{a}'\boldsymbol{d} = \boldsymbol{p}_2'\boldsymbol{d}$. This gives us an infinite line. Most practical applications require a finite segment. The endpoints can be estimated by (1) finding the points on the line that lie close to observed points in both planes and then (2) finding the two extremes of those points. On the other hand, finding straight 3-D lines can be easier with a stereo sensor, as these result from matching two straight two-dimensional (2-D) image lines.

Triangulated Surfaces

The goal of the process is to estimate a triangulated mesh from a set of 3-D data points. If the points are sampled on a regular grid (e.g., from a $2\frac{1}{2}$-D range image), then the triangles can be formed naturally from the sampling (Fig. 22.13). If the points are part of a 3-D point set, then triangulation is harder. We do not give details here because of the complexity of this task, but some of the issues that have to be considered are: (1) how to find a surface that lies close to all data points (as it is unlikely to actually pass through all because of noise), (2) how to avoid holes and wild triangles in the mesh, (3) how to choose the threshold of closeness so that only one surface passes through sections of a point cloud, (4) how to choose the triangle size, (5) what to do about outlying points, and (6) how to maintain observable scene features, such as surface edges. Popular early approaches for triangulation are the marching cube and triangle [22.34, 35] algorithms. These algorithms can give meshes with many triangles. The triangle size and thus the computational complexity of using the mesh can be reduced by mesh decimation, for which many algorithms exist [22.36, 37].

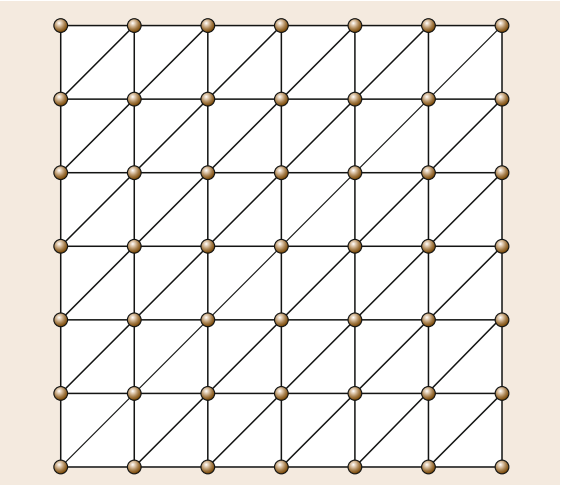

Fig. 22.13 Triangulation of a regular grid of points

22.2.3 Model Matching and Multiple-View Registration

Model matching is the process of matching some stored representation to some observed data. In the case discussed here, we assume that both are 3-D representations. Furthermore, we assume that the representations being matched are both of the same type, e.g., 3-D model and scene lines. (While different types of data can also be matched, we ignore these more specialized algorithms here.)

A special case of matching is when the two structures being matched are both scene or model surfaces. This commonly occurs when attempting to produce a larger representation of the scene by fusing multiple partial views of the scene. These partial views can be acquired as a mobile vehicle navigates around the scene, observing it from different places. The mobile robotics simultaneous localization and mapping (SLAM) algorithm [22.38] (discussed in Chap. 37) incrementally fuses newly observed scene structure to its growing scene map model, while also matching portions of that newly observed data to previously observed data to estimate its current location. One SLAM project [22.39] used scale invariant feature transform (SIFT) feature points for 3-D modeling and map construction.

The algorithm used for matching depends on the complexity the structures being matched. If the structures being matched are extended geometric entities such as planes or 3-D lines, then a discrete matching algorithm like the interpretation tree algorithm [22.40] can be used. Alternatively, if the structures are low level, such as 3-D points or triangles, then a point set alignment algorithm like the iterated closest point (ICP) algorithm [22.41] can be used. Both of these are described in more detail below.

After structures have been matched, the commonest next step is to estimate the transformation (rotation plus translation) linking the two datasets. This process is described in the next section. With the transforma-

tion, we can then transform the data into the same coordinate system and fuse (combine) the datasets. If the structures being fused are points, then the result is a larger point set formed from the combination of the two (or more) original datasets. Sets of planes and lines can also be simply added together, subject to representing the matched members of the two datasets only once. Triangulated surfaces require a more complex merging, in that we require a topologically correct triangulated mesh to be the result. The zippering algorithm [22.42] is a well-known triangulated mesh merging algorithm.

The interpretation tree algorithm [22.40] (see Table 22.2) is suitable for matching small numbers (e.g., less than about 20–30) of discrete objects, such as vertical edges seen in 2-D or 3-D. If there are M model and D data objects, then potentially there are M^D different matches. The key to efficient matching is to identify pairwise constraints that eliminate unsuitable matches. Constraints between pairs of model features and pairs of data features also greatly reduce the matching space. If the constraints eliminate enough features, a polynomial-time algorithm results. The core of the algorithm is defined as follows. Let $\{m_i\}$ and $\{d_j\}$ be the sets of model and data features to be matched, $u(m_i, d_j)$ is true if m_i and d_j are compatible features, $b(m_i, m_j, d_k, d_l)$ is true if the four model and data features are compatible, and T is the minimum number of matched features before a successful match is declared. `Pairs` is the set of successfully matched features. The function `truesizeof` counts the number of actual matches in the set, disregarding matches with the wildcard '*' which matches anything.

When the data being matched are sets of points or triangles, then the ICP algorithm [22.41] is more commonly used. This algorithm estimates the pose transformation that aligns the two sets to a minimum distance. With this transformation, the two sets can be represented in the same coordinate system and treated as a larger single set (perhaps with some merging of overlapping data).

Here we give the algorithm for point matching, but this can be easily adapted for other sets of features. This algorithm is iterative, so may not always converge quickly, but often several iterations is sufficient. However, ICP can produce a bad alignment of the data sets. Best results arise with a good initial alignment, such as from odometry or previous positions. The ICP algorithm can be extended to include other properties when matching, such as color or local neighborhood structure. A good spatial indexing algorithm (e.g., k-d trees) is necessary for efficient matching in the closest point function (CP) below.

Let $\mathcal{S}$ be a set of N_s points $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{N_s}\}$ and $\mathcal{M}$ be the model. Let $||\boldsymbol{s} - \boldsymbol{m}||$ be the Euclidean distance between point $\boldsymbol{s} \in \mathcal{S}$ and $\boldsymbol{m} \in \mathcal{M}$. Let CP$(\boldsymbol{s}, \mathcal{M})$ be the closest point (Euclidean distance) in $\mathcal{M}$ to the scene point $\boldsymbol{s}$.

1. Let $\boldsymbol{T}^{[0]}$ be an initial estimate of the rigid transformation aligning the two sets.
2. Repeat for $k = 1 \ldots k_{\max}$ or until convergence:
   a) Compute the set of correspondences $\mathcal{C} = \bigcup_{i=1}^{N_s} \{(\boldsymbol{s}_i, \text{CP}(\boldsymbol{T}^{[k-1]}(\boldsymbol{s}_i), \mathcal{M}))\}$.

Table 22.2 Interpretation tree algorithm

```
pairs=it(0,{})
if truesizeof(pairs) >= T, then success

function pairs=it(level,inpairs)
  if M-level+truesizeof(inpairs) < T
    then return {} % can never succeed
  for each d_i % loopD start
    if not u(m_level,d_i), then continue loopD
    for each (m_k,d_l) in inpairs
      if not b(m_level,m_k,d_i,d_l)
        then continue loopD
    endfor    % have found a successful new pair to add
    pairs = it(level+1, union(inpairs,(m_level,d_i)))
    if truesizeof(pairs) >= T, then return pairs
  endfor % loopD end
  % no success, so try wildcard
  pairs = it(level+1,union(inpairs,(m_level,*)))
```

b) Compute the new Euclidean transformation $\boldsymbol{T}^{[k]}$ that minimizes the mean square error between point pairs in $\mathcal{C}$.

22.2.4 Maximum-Likelihood Registration

One popular technique for range scan matching, especially in the 2-D case, uses maximum likelihood as the criterion for match goodness. This technique also takes into account a probabilistic model of the range readings. Let r be a range reading of a sensor scan at position s, and $\bar{r}$ the distance to the nearest object along the path of r (which depends on s). Then a model of the sensor reading is the probability $p(r|\bar{r})$. Typically the model has a Gaussian shape around the correct range, with an earlier false-positive region and a missed-reading region at the end (Fig. 22.14).

A scan is matched when its position produces a maximum-likelihood result for all the readings, based on a reference scan. Assuming independence of the readings, from Bayes' rule we get

$$\max_s p(s|\boldsymbol{r}) = \max_s \prod_i p(r_i|\bar{r}_i) \,.$$

The maximum likelihood scan location can be found by hill-climbing or exhaustive search from a good starting point. Figure 22.15 shows the results of maximum likelihood for aligning a scan against several reference scans. In this 2-D case, the reference scans are put into an occupancy grid for computing $\bar{r}$ [22.44]. A widespread modification for efficiency is to compute $\bar{r}$ by a smearing operation on the occupancy grid, ignoring line-of-sight

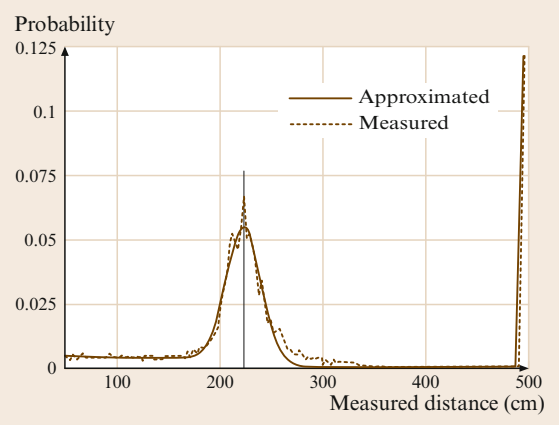

Fig. 22.14 Probability profile $p(r|\bar{r})$ for a laser sensor reading. The Gaussian peak occurs at the distance $\bar{r}$ of the obstacle [22.43]

Fig. 22.15 Scan match by maximum likelihood against several reference scans in an occupancy grid [22.44]

information [22.45]. In the 3-D case, triangulated surfaces are more appropriate than voxels [22.46].

22.2.5 Multiple-Scan Registration

In the previous subsection, multiple reference scans could be used to form an occupancy grid or surface triangles for matching. In general, scan matching between sets of scans produces a network of constraints among the scans, for example, in going around a loop, successive scans form a chain of constraints, and the first and last scans form a closed loop of constraints. Globally consistent registration of scans is a part of SLAM (Chap. 37).

If the individual constraints have covariance estimates, then maximum-likelihood techniques can be used to find a globally consistent estimate for all the scans [22.47]. This global registration is among the scan *poses*, and does not involve the scans themselves – all information has been abstracted to constraints among the poses. Let $\bar{s}_{ij}$ be the scan-matched pose difference between scans s_i and s_j, with covariance Γ_{ij}. The maximum-likelihood estimate for all s is given by the nonlinear least-squares system

$$\min_s \sum_{ij} (s_{ij} - \bar{s}_{ij})^\top \Gamma_{ij} (s_{ij} - \bar{s}_{ij}) \,, \qquad (22.8)$$

which can be solved efficiently using iterative least-squares techniques, e.g., Levenberg–Marquadt or conjugate gradient methods [22.48, 49].

A complication is that the system of constraints is calculated based on an initial set of poses s. Given a repositioning of s from (22.8), redoing the scan matching will give a different set of constraints, in general. Iterating the registration process with new constraints is not guaranteed to lead to a global minimum; in practice, with a good initial estimate, convergence is often rapid and robust.

22.2.6 Relative Pose Estimation

Central to many tasks is the estimation of the coordinate system relative position or pose transformation between

two coordinate systems, for example, this might be the pose of a scanner mounted on a mobile vehicle relative to scene landmarks, or it might be the relative pose of some scene features as observed in two views taken from different positions.

We present here three algorithms that cover most instances of the pose estimation process, which differ slightly based on the type of feature being matched.

Point Set Relative Pose Estimation

The first algorithm matches point sets. Assume that N 3-D points $\{\boldsymbol{m}_i\}$ in one structure or coordinate system are matched to points $\{\boldsymbol{d}_i\}$ in a different coordinate system. The matching might be found as part of the alignment process in the ICP algorithm just described or they might be observed triangle vertices matched to a previously constructed triangulated scene model, for example. The desired pose is the rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ that minimizes $\sum_i \| \boldsymbol{R}\boldsymbol{m}_i + \boldsymbol{t} - \boldsymbol{d}_i \|^2$. Compute the mean vectors $\boldsymbol{\mu}_m$ and $\boldsymbol{\mu}_d$ of the two sets. Compute the centralized point sets by $\boldsymbol{a}_i = \boldsymbol{m}_i - \boldsymbol{\mu}_m$ and $\boldsymbol{b}_i = \boldsymbol{d}_i - \boldsymbol{\mu}_d$. Construct the $3 \times N$ matrix $\boldsymbol{A}$ that consists of the vectors $\{\boldsymbol{a}_i\}$ stacked up. Construct the $3 \times N$ matrices $\boldsymbol{B}$ in a similar way from the vectors $\{\boldsymbol{b}_i\}$. Compute the singular value decomposition $\text{svd}(\boldsymbol{B}\boldsymbol{A}') = \boldsymbol{U}'\boldsymbol{D}\boldsymbol{V}'$ [22.50]. Compute the rotation matrix $\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}'$. If the data points are nearly planar then this calculation can introduce a mirror-image transformation. This can be checked with the vector triple product and, if mirrored, the diagonal correction matrix C(1,1,-1) can be used in $\boldsymbol{R} = \boldsymbol{V}\boldsymbol{C}\boldsymbol{U}'$. Finally, compute the translation vector $\boldsymbol{t} = \boldsymbol{\mu}_d - \boldsymbol{R}\boldsymbol{\mu}_m$. With these least-squares estimates of the transformation, a point $\boldsymbol{m}_i$ transforms to point $\boldsymbol{R}\boldsymbol{m}_i + \boldsymbol{t}$, which should be near to point $\boldsymbol{d}_i$.

Straight Line Relative Pose Estimation

If 3-D lines are the features that are extracted, then the relative pose transformation can be estimated as follows. Assume N paired lines. The first set of lines is described by direction vectors $\{\boldsymbol{e}_i\}$ and a point on each line $\{\boldsymbol{a}_i\}$. The second set of lines is described by direction vectors $\{\boldsymbol{f}_i\}$ and a point on each line $\{\boldsymbol{b}_i\}$. In this algorithm, we assume that the direction vectors on the matched segments always point the same direction (i. e., they are not inverted). This can be achieved by exploiting some scene constraints, or trying all combinations and eliminating inconsistent solutions. The points $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ need not correspond to the same point after alignment. The desired rotation matrix $\boldsymbol{R}$ minimizes $\sum_i \| \boldsymbol{R}\boldsymbol{e}_i - \boldsymbol{f}_i \|^2$. Construct the $3 \times N$ matrices $\boldsymbol{E}$ that consist of the vectors $\{\boldsymbol{e}_i\}$ stacked up. Construct the $3 \times N$ matrices $\boldsymbol{F}$ in a similar way from the vectors $\{\boldsymbol{f}_i\}$. Compute the singular value decomposition $\text{svd}(\boldsymbol{F}\boldsymbol{E}') = \boldsymbol{U}'\boldsymbol{D}\boldsymbol{V}'$. Compute the rotation matrix $\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}'$. The translation estimate $\boldsymbol{t}$ minimizes the sum of the square of the distances λ_i between the rotated points $\boldsymbol{a}_i$ and corresponding line $(\boldsymbol{f}_i, \boldsymbol{b}_i)$. Define the matrix $\boldsymbol{L} = \sum_i (\boldsymbol{I} - \boldsymbol{f}_i \boldsymbol{f}_i')'(\boldsymbol{I} - \boldsymbol{f}_i \boldsymbol{f}_i')$. Define the vector $\boldsymbol{n} = \sum_i (\boldsymbol{I} - \boldsymbol{f}_i \boldsymbol{f}_i')'(\boldsymbol{I} - \boldsymbol{f}_i \boldsymbol{f}_i')(\boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{b}_i)$. Then the translation is $\boldsymbol{t} = -\boldsymbol{L}^{-1}\boldsymbol{n}$.

Plane Relative Pose Estimation

Finally, if planes are the 3-D features extracted for matching, then the relative pose transformation can be estimated as follows. Assume N paired planes. The first set of planes is described by surface normals $\{\boldsymbol{e}_i\}$ and a point on each plane $\{\boldsymbol{a}_i\}$. The second set of planes is described by surface normals $\{\boldsymbol{f}_i\}$ and a point on each plane $\{\boldsymbol{b}_i\}$. Here we assume that the surface normals always point outward from the surface. The points $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ need not correspond to the same point after alignment. The desired rotation matrix $\boldsymbol{R}$ minimizes $\sum_i \| \boldsymbol{R}\boldsymbol{e}_i - \boldsymbol{f}_i \|^2$. Construct the $3 \times N$ matrices $\boldsymbol{E}$ that consist of the vectors $\{\boldsymbol{e}_i\}$ stacked up. Construct the $3 \times N$ matrices $\boldsymbol{F}$ in a similar way from the vectors $\{\boldsymbol{f}_i\}$. Compute the singular value decomposition $\text{svd}(\boldsymbol{F}\boldsymbol{E}') = \boldsymbol{U}'\boldsymbol{D}\boldsymbol{V}'$. [22.50] Compute the rotation matrix $\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}'$. The translation estimate $\boldsymbol{t}$ minimizes the sum of the square of the distances λ_i between the rotated point $\boldsymbol{a}_i$ and the corresponding plane $(\boldsymbol{f}_i, \boldsymbol{b}_i)$. Define the matrix $\boldsymbol{L} = \sum_i \boldsymbol{f}_i \boldsymbol{f}_i'$. Define the vector $\boldsymbol{n} = \sum_i \boldsymbol{f}_i \boldsymbol{f}_i'(\boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{b}_i)$. Then the translation is $\boldsymbol{t} = -\boldsymbol{L}^{-1}\boldsymbol{n}$.

In all of the calculations described above, we assumed normally distributed errors. For techniques to make these sorts of calculations more robust, see *Zhang* [22.51].

22.2.7 3-D Applications

This section links the techniques presented above to the robotics applications of 3-D localization of parts for robot manipulation, self-localization of robot vehicles, and scene understanding for robot navigation. The robotics tasks mentioned here are discussed in more detail in other chapters in the series. While this chapter focuses on robotics applications, there are many other 3-D sensing applications. An area of much current research is that of acquiring 3-D models, particularly for reverse engineering of mechanical parts [22.52], historical artifacts [22.53], buildings [22.54], and people for computer games and movies (see, e.g., Cyberware's Whole-Body X 3-D scanner).

The key tasks in robot manipulation are: (1) identification of grasping points (Chaps. 27 and 28),

(2) identification of a collision-free grasp (Chaps. 27 and 28), (3) recognition of parts to be manipulated (Chap. 23) and (4) position estimation of parts for manipulation (Chaps. 23 and 42).

The key tasks in robot navigation and self-localization are: (5) identification of a navigable ground plane (Sect. 22.3), (6) identification of a collision-free path (Chap. 35), (7) identification of landmarks (Chap. 36), and (8) estimation of vehicle location (Chaps. 37 and 40).

The mobile and assembly robotics tasks link together rather naturally. Tasks 1 and 5 have a connection, when we consider these tasks in the context of unknown parts or paths. Part grasping requires finding regions on a part that are graspable, which usually means locally planar patches that are large enough that a gripper can make good contact with them. Similarly, navigation usually requires smooth ground regions that are large enough for the vehicle – again locally planar patches. Both tasks are commonly based on triangulated scene methods to represent the data, from which connected regions of nearly coplanar patches can be extracted. The main difference between these two tasks is the ground-plane detection task is looking for a larger patch, that must be on the *ground* and upward facing.

Tasks 2 and 6 require a method of representing empty space along the proposed trajectory of the gripper contacts or the vehicle. The voxel representation is good for this task.

Tasks 3 and 7 are model matching tasks and can use the methods of Sect. 22.2.3 to match observed scene features to prestored models of known parts or scene locations. Commonly used features are large planar surfaces, 3-D edges, and 3-D feature points.

Tasks 4 and 8 are pose estimation tasks and can use the methods of Sect. 22.2.6 to estimate the pose of the object relative to the sensor or vehicle (i. e., sensor) relative to the scene. Again, commonly used features are large planar surfaces, 3-D edges, and 3-D feature points.

22.3 Navigation and Terrain Classification

One of the more compelling uses for range data is for navigation of mobile robot vehicles. Range data provides information about obstacles and free space for the vehicle, in a direct geometric form. Because of the real-time constraints of navigation, it is often impractical to reconstruct a full 3-D model of the terrain using the techniques presented in this Chapter. Instead, most systems use an *elevation model*. An elevation model is a tesselated 2-D representation of space, where at each cell there is information about the distribution of 3-D points in the cell. In its simplest incarnation, the elevation map just contains the mean height of range points above the nominal ground plane (Fig. 22.16). This representation is sufficient for some indoor and urban environments; more sophisticated versions that determine a local plane, scatter of points in the cell, etc., are useful for more complicated off-road driving. Elevation maps marked with obstacles have obvious utility for planning a collision-free path for the vehicle.

22.3.1 Indoor Reconstruction

SLAM algorithms (Chap. 37) using 2-D laser range finders can reconstruct floor plans with centimeter precision. Some research has extended this work to 3-D reconstruction, using 2-D lasers that are swept along as the robot moves [22.56]. The resultant point cloud is typically registered using the pose of the robot as corrected by the 2-D SLAM algorithm, rather than any of the 3-D registration techniques covered in this chapter, because the laser is swept using robot motion.

The raw points can be presented as a 3-D image, or triangulated to give a planar or mesh reconstruction of indoor surfaces. The latter is especially compelling when camera images are texture-mapped onto the surfaces, creating a realistic 3-D model. Figure 22.17 shows some results from indoor mapping using this technique.

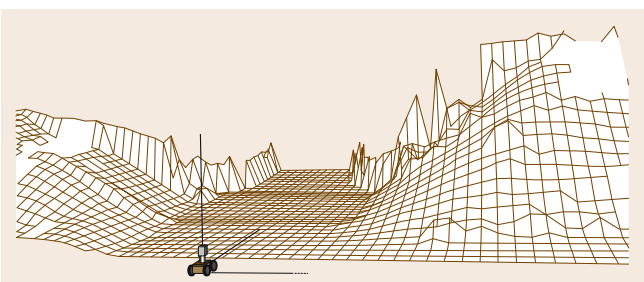

Fig. 22.16 Elevation map in urban terrain. Each cell holds the height of the terrain at that point. More extensive features can also be incorporated: slope, point variance, etc. [22.55]

Fig. 22.17 3-D indoor map from a swept vertical plane LADAR. Registration is from a horizontal LADAR using SLAM algorithms [22.56]

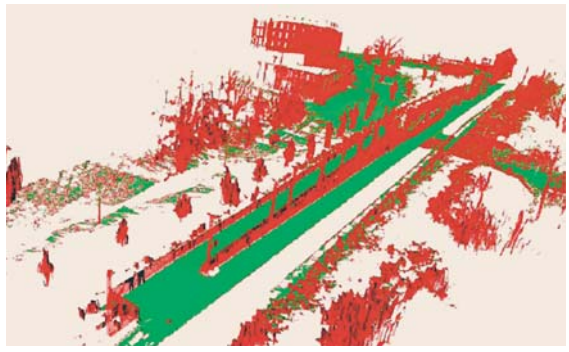

Fig. 22.18 Elevation map of an urban scene, using 10 cm × 10 cm cells. Obstacles in *red*, ground plane in *green* [22.55]

Smoothing of surface facets can be used to recover planar surfaces [22.57].

22.3.2 Urban Navigation

In urban navigation, the environment is structured, with roads, buildings, sidewalks, and also moving objects – people and other vehicles. There are two main challenges: how to register laser scans from a fast-moving vehicle for consistent mapping, and how to detect moving objects using range scans (of course, other methods are also used for detecting moving objects, e.g., appearance-based vision).

Outdoor vehicles can use precision global positioning system (GPS), inertial measurement units, and wheel odometry to keep track of their position and orientation, typically with an extended Kalman filter. This method is good enough to obviate the need for precise registration matching among scans, as long as the motion model of the vehicle, and timing from the range scanner, is used to place each scan reading in its proper position in the world model. This method also works in relatively easy off-road terrain such as in the Defense Advanced Research Projects Agency (DARPA) Grand Challenge [22.59]. In all cases, the reduction of pose estimation error is critical for good performance [22.60].

Once scan readings are registered using the vehicle pose estimation, they can be put into an elevation map, and obstacles detected using the slope and vertical extent of the range readings in the cells of the map. A complication is that there may be multiple levels of elevation in an urban setting, for example, an overpass would not be an obstacle if it were high enough. One proposal is to use multiple elevation clusters within each cell; this technique is called a *multilevel surface map* (MLS, [22.55]). Each cell in the map stores a set of surfaces represented by a mean height and variance. Figure 22.18 shows an MLS with a cell size of 10 cm^2, with ground plane and obstacles marked.

For dynamic objects, real-time stereo at 15–30 Hz can capture the motion of the objects. When the stereo rig is fixed, range background subtraction isolates just

Fig. 22.19a–c Independent motion detection from a moving platform. The reference image **(a)** is forward-projected using the motion homography to **(b)**; **(c)** is the difference with the actual image [22.58]

the moving objects [22.61]. When the rig is on a moving vehicle, the problem is more difficult, since the whole scene is moving with respect to the rig. It can be solved by estimating the motion of the rig with respect to the dominant rigid background of the scene. Let R, t be the motion of the rig between two frames, estimated by extracting features and matching them across the two temporal frames, using the techniques of Chap. 37. The homography $H(R, t)$ of (22.6) provides a direct projection of the disparity vectors $p_0 = (x_0, y_0, d_0, 1)$ of the first frame to their correspondences $H(R, t)p_0$ under R, t in the second frame. Using the homography allows the points in the reference frame to be directly projected onto the next frame, without translating to 3-D points. Figure 22.19 shows the projected pixels under rigid motion from a reference scene. The difference between the projected and actual pixels gives the independently moving objects [22.58].

22.3.3 Rough Terrain

Rough outdoor terrain presents two challenges:

- There may be no extensive ground plane to characterize driveability and obstacles.
- Vegetation that is pliable and driveable may appear as an obstacle in range images.

Figure 22.20 shows a typical outdoor scene, with a small (1 m) robot driving through vegetation and rough ground [22.62]. Range data from stereo vision on the robot will see the top of the vegetation and some ground points below. The elevation model can be extended to look at *point statistics* within each cell, to capture the notion of a local ground plane and penetrability related to vegetation. In [22.65], for example, the set of proposed features includes:

- major plane slope using a robust fit (Sect. 22.2.2)
- height difference of maximum and minimum heights
- points above the major plane
- density: the ratio of points in the cell to rays that pass through the cell

The density feature is interesting (and expensive to compute), and attempts to characterize vegetation such a grass or bushes, by looking at whether range readings penetrate an elevation cell. The idea of using range readings to estimate vegetation has been discussed in several other projects on off-road driving [22.64, 66, 67].

Elevation map cells can be characterized as obstacles or driveable through learning or hand-built classifiers. Among the learning techniques are neural nets [22.65] and Gaussian mixture models with expectation-maximization learning [22.63]. The latter work also includes a lower level of interpretation, classifying surfaces into planar patches (ground plane, solid obstacles), linear features (telephone wires), and scattered features (vegetation). Figure 22.21 shows some results from a laser-scanned outdoor scene. Linear features such as telephone wires and the telephone pole are accurately determined, as well as vegetation with high penetrability.

Fig. 22.20 Rough terrain, no ground plane, driveable vegetation [22.62]

Some additional problems occur in rough-terrain navigation. For planar laser range finders that are swept over the terrain by vehicle motion, the precision of vehicle pose estimation is important for accurate reconstruction. Attitude errors of less than 0.5° can cause false positives in obstacle detection, especially for sweeps far

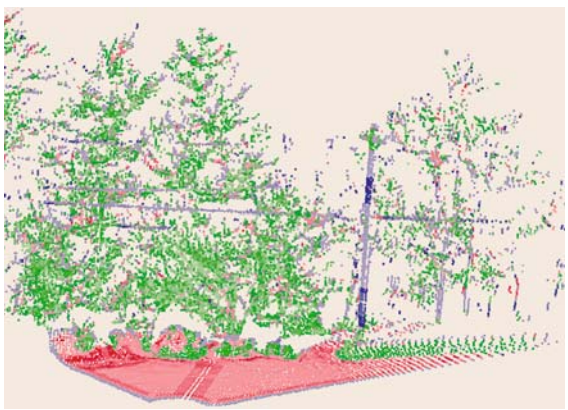

Fig. 22.21 Classification using point statistics, from [22.63]. *Red* is planar surface, *blue* is thin linear surface, *green* is scattered penetrable surface [22.64]

ahead of the vehicle. In [22.60], this problem is solved by looking at the time of each laser reading, and noting a correlation between height errors and time difference in the readings.

Negative obstacles (ditches and cliffs) are difficult to detect with range information, because the sensor may not see the bottom of the obstacle. This is especially true for vehicle-mounted sensors that are not very high off the ground, and that are looking far ahead. Negative obstacles can be inferred when there is a gap in the ground plane, and a plane slanted upwards at the back edge of the gap. Such artifacts can be efficiently found using column search on the disparity image [22.68].

22.4 Conclusions and Further Reading

Range sensing is an active and expanding field of research in robotics. The presence of new types of devices – flash LADARs, multi-beam LADARs, on-camera stereo processing – and the continuing development of robust algorithms for object reconstruction, localization, and mapping has helped to bring applications out of the laboratory and into the real world. Indoor navigation with LADARs is already being exploited in commercial products (see, for example, [22.69]). As the basic capabilities become more robust, researchers are looking to perform useful tasks, such as fetching items or doing dishes [22.70].

Another set of challenges are found in less benign environments, such as urban and off-road driving (DARPA Grand Challenge and Urban Challenge [22.59]). Stereo vision and laser range finding also will play a role in helping to provide autonomy for a new generation of more capable robotic platforms that rely on walking for locomotion [22.71]. The challenges are dealing with motion that is less smooth than wheeled platforms, environments that contain dynamic obstacles, and task-oriented recognition of objects.

References

22.1 Videre Design LLC: www.videredesign.com, accessed Nov 12, 2007 (Videre Design, Menlo Park 2007)

22.2 R. Hartley, A. Zisserman: *Multiple view geometry in computer vision* (Cambridge Univ. Press, Cambridge 2000)

22.3 S. Barnard, M. Fischler: Computational stereo, ACM Comput. Surv. **14**(4), 553–572 (1982)

22.4 K. Konolige: Small vision system. hardware and implementation, Proc. Int. Symp. Robot. Res. (Hayama 1997) pp. 111–116

22.5 D. Scharstein, R. Szeliski, R. Zabih: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, Int. J. Comput. Vis. **47**(1/2/3), 7–42 (2002)

22.6 D. Scharstein, R. Szeliski: Middlebury College Stereo Vision Research Page, vision.middlebury.edu/stereo, accessed Nov 12, 2007 (Middleburry College, Middleburry 2007)

22.7 R. Zabih, J. Woodfill: Non-parametric local transforms for computing visual correspondence, Proc. Eur. Conf. on Computer Vision, Vol. 2 (Stockholm 1994) pp. 151–158

22.8 O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, C. Proy: Real time correlation based stereo: algorithm implementations and applications, Tech. Report RR-2013, INRIA (1993)

22.9 M. Okutomi, T. Kanade: A multiple-baseline stereo, IEEE Trans. Patt. Anal. Mach. Intell. **15**(4), 353–363 (1993)

22.10 L. Matthies: Stereo vision for planetary rovers: stochastic modeling to near realtime implementation, Int. J. Comput. Vis. **8**(1), 71–91 (1993)

22.11 R. Bolles, J. Woodfill: Spatiotemporal consistency checking of passive range data, Proc. Int. Symp. on Robotics Research (Hidden Valley 1993)

22.12 P. Fua: A parallel stereo algorithm that produces dense depth maps and preserves image features, Mach. Vis. Appl. **6**(1), 35–49 (1993)

22.13 H. Moravec: Visual mapping by a robot rover, Proc. Int. Joint Conf. on AI (IJCAI) (Tokyo 1979) pp. 598–600

22.14 A. Adan, F. Molina, L. Morena: Disordered patterns projection for 3D motion recovering, Proc. Int. Conf. on 3D Data Processing, Visualization and Transmission (Thessaloniki 2004) pp. 262–269

22.15 Point Grey Research Inc.: www.ptgrey.com, accessed Nov 12, 2007 (Point Grey Research, Vancouver 2007)

22.16 C. Zach, A. Klaus, M. Hadwiger, K. Karner: Accurate dense stereo reconstruction using graphics

hardware, Proc. EUROGRAPHICS (Granada 2003) pp. 227–234
22.17 R. Yang, M. Pollefeys: Multi-resolution real-time stereo on commodity graphics hardware, Int. Conf. Computer Vision and Pattern Recognition, Vol. 1 (Madison 2003) pp. 211–217
22.18 Focus Robotics Inc.: www.focusrobotics.com, accessed Nv 12, 2007 (Focus Robotics, Hudson 2007)
22.19 TYZX Inc.: www.tyzx.com, accessed Nov 12, 2007 (TYZX, Menlo Park 2007)
22.20 S.K. Nayar, Y. Nakagawa: Shape from focus, IEEE Trans. Patt. Anal. Mach. Intell. **16**(8), 824–831 (1994)
22.21 M. Pollefeys, R. Koch, L. Van Gool: Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters, Int. J. Comput. Vis. **32**(1), 7–25 (1999)
22.22 A. Hertzmann, S.M. Seitz: Example-based photometric stereo: Shape reconstruction with general, varying BRDFs, IEEE Trans. Patt. Anal. Mach. Intell. **27**(8), 1254–1264 (2005)
22.23 A. Lobay, D.A. Forsyth: Shape from texture without boundaries, Int. J. Comput. Vis. **67**(1), 71–91 (2006)
22.24 F. Blais: Review of 20 years of range sensor development, J. Electron. Imag. **13**(1), 231–240 (2004)
22.25 R. Baribeau, M. Rioux, G. Godin: Color reflectance modeling using a polychromatic laser range sensor, IEEE Trans. Patt. Anal. Mach. Intell. **14**(2), 263–269 (1992)
22.26 D. Anderson, H. Herman, A. Kelly: Experimental Characterization of Commercial Flash Ladar Devices, Int. Conf. of Sensing and Technology (Palmerston North 2005) pp. 17–23
22.27 R. Stettner, H. Bailey, S. Silverman: Three-Dimensional Flash Ladar Focal Planes and Time-Dependent Imaging, Advanced Scientific Concepts, 2006; Technical Report (February 23, 2007): www.advancedscientificconcepts.com/images/Three Dimensional Flash Ladar Focal Planes-ISSSR Paper.pdf, accessed Nov 12, 2007 (Advanced Scientific Concepts, Santa Barbara 2007)
22.28 J.J. LeMoigne, A.M. Waxman: Structured light patterns for robot mobility, Robot. Autom. **4**, 541–548 (1988)
22.29 R.B. Fisher, D.K. Naidu: A Comparison of Algorithms for Subpixel Peak Detection. In: *Image Technology*, ed. by J. Sanz (Springer, Berlin, Heidelberg 1996)
22.30 J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes: *Computer Graphics: principles and practice* (Addison Wesley, Reading 1996)
22.31 B. Curless, M. Levoy: A Volumetric Method for Building Complex Models from Range Images, Proc. of Int. Conf. on Comput. Graph. and Inter. Tech. (SIGGRAPH) (New Orleans 1996) pp. 303–312
22.32 A. Hoover, G. Jean-Baptiste, X. Jiang, P.J. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, R. Fisher: An experimental comparison of range segmentation algorithms, IEEE Trans. Patt. Anal. Mach. Intell. **18**(7), 673–689 (1996)
22.33 M.A. Fischler, R.C. Bolles: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM **24**(6), 381–395 (1981)
22.34 H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle: Surface reconstruction from unorganized points, Comput. Graph. **26**(2), 71–78 (1992)
22.35 A. Hilton, A. Stoddart, J. Illingworth, T. Windeatt: Implicit surface-based geometric fusion, Comput. Vis. Image Under. **69**(3), 273–291 (1998)
22.36 H. Hoppe: New quadric metric for simplifying meshes with appearance attributes, IEEE Visualization 1999 Conference (San Francisco 1999) pp. 59–66
22.37 W.J. Schroeder, J.A. Zarge, W.E. Lorensen: Decimation of triangle meshes, Proc. of Int. Conf. on Comput. Graph. and Inter. Tech. (SIGGRAPH) (Chicago 1992) pp. 65–70
22.38 S. Thrun: A probabilistic online mapping algorithm for teams of mobile robots, Int. J. Robot. Res. **20**(5), 335–363 (2001)
22.39 J. Little, S. Se, D. Lowe: Vision based mobile robot localization and mapping using scale-invariant features, Proc. IEEE Inf. Conf. on Robotics and Automation (Seoul 2001) pp. 2051–2058
22.40 E. Grimson: *Object Recognition by Computer: The Role of Geometric Constraints* (MIT Press, London 1990)
22.41 P.J. Besl, N.D. McKay: A method for registration of 3D shapes, IEEE Trans. Patt. Anal. Mach. Intell. **14**(2), 239–256 (1992)
22.42 G. Turk, M. Levoy: Zippered Polygon Meshes from Range Images, Proc. of Int. Conf. on Comput. Graph. and Inter. Tech. (SIGGRAPH) (Orlando 1994) pp. 311–318
22.43 S. Thrun, W. Burgard, D. Fox: *Probabilistic Robotics* (MIT Press, Cambridge 2005)
22.44 D. Haehnel, D. Schulz, W. Burgard: Mapping with mobile robots in populated environments, Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Vol. 1 (Lausanne 2002) pp. 496–501
22.45 K. Konolige, K. Chou: Markov localization using correlation, Proc. Int. Joint Conf. on AI (IJCAI) (Stockholm 1999) pp. 1154–1159
22.46 D. Haehnel, W. Burgard: Probabilistic Matching for 3D Scan Registration, Proc. of the VDI-Conference Robotik 2002 (Robotik) (Ludwigsburg 2002)
22.47 F. Lu, E. Milios: Globally consistent range scan alignment for environment mapping, Auton. Robot. **4**, 333–349 (1997)
22.48 K. Konolige: Large-scale map-making, Proceedings of the National Conference on AI (AAAI) (San Jose 2004) pp. 457–463
22.49 A. Kelly, R. Unnikrishnan: Efficient Construction of Globally Consistent Ladar Maps using Pose Network

Topology and Nonlinear Programming, Proc. Int. Symp of Robotics Research (Siena 2003)
22.50 K.S. Arun, T.S. Huang, S.D. Blostein: Least-squares fitting of two 3-D point sets, IEEE Trans. Patt. Anal. Mach. Intell. **9**(5), 698–700 (1987)
22.51 Z. Zhang: Parameter estimation techniques: a tutorial with application to conic fitting, Image Vis. Comput. **15**, 59–76 (1997)
22.52 P. Benko, G. Kos, T. Varady, L. Andor, R.R. Martin: Constrained fitting in reverse engineering, Comput. Aided Geom. Des. **19**, 173–205 (2002)
22.53 M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, D. Fulk: The Digital Michelangelo Project: 3D Scanning of Large Statues, Proc. 27th Conf. on Computer graphics and interactive techniques (SIGGRAPH) (New Orleans 2000) pp. 131–144
22.54 I. Stamos, P. Allen: 3-D Model Construction Using Range and Image Data, Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Vol. 1 (Hilton Head Island 2000) pp. 531–536
22.55 R. Triebel, P. Pfaff, W. Burgard: Multi-level surface maps for outdoor terrain mapping and loop closing, Proc. of the IEEE Int. Conf. on Intel. Robots and Systems (IROS) (Beijing 2006)
22.56 S. Thrun, W. Burgard, D. Fox: A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping, Proc. IEEE Inf. Conf. on Robotics and Automation (San Francisco 2000) pp. 321–328
22.57 Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, S. Thrun: Using EM to Learn 3D Models of Indoor Environments with Mobile Robots, Proc.Int. Conf. on Machine Learning (Williamstown 2001) pp. 329–336
22.58 M. Agrawal, K. Konolige, L. Iocchi: Real-time detection of independent motion using stereo, IEEE Workshop on Motion (Breckenridge 2005) pp. 207–214
22.59 The DARPA Grand Challenge: www.darpa.mil/grandchallenge05, accessed Nov 12, 2007 (DARPA, Arlington 2005)
22.60 S. Thrun, M. Montemerlo, H. Dahlkamp et al.: Stanley: The robot that won the DARPA Grand Challenge, J. Field Robot. **23**(9), 661–670 (2006)
22.61 C. Eveland, K. Konolige, R. Bolles: Background modeling for segmentation of video-rate stereo sequences, Proc. Int. Conf. on Computer Vision and Pattern Recog (Santa Barbara 1998) pp. 266–271
22.62 K. Konolige, M. Agrawal, R.C. Bolles, C. Cowan, M. Fischler, B. Gerkey: Outdoor mapping and Navigation using Stereo Vision, Intl. Symp. on Experimental Robotics (ISER) (Rio de Janeiro 2006)
22.63 J. Lalonde, N. Vandapel, D. Huber, M. Hebert: Natural terrain classification using three-dimensional ladar data for ground robot mobility, J. Field Robot. **23**(10), 839–861 (2006)
22.64 J.-F. Lalonde, N. Vandapel, M. Hebert: Data structure for efficient processing in 3-D, Robotics: Science and Systems 1, Cambridge (2005)
22.65 M. Happold, M. Ollis, N. Johnson: enhancing supervised terrain classification with predictive unsupervised learning, Robotics: Science and Systems (Philadelphia 2006)
22.66 R. Manduchi, A. Castano, A. Talukder, L. Matthies: Obstacle detection and terrain classification for autonomous off-road navigation, Auton. Robot. **18**, 81–102 (2005)
22.67 A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, R. Warner: Toward reliable off road autonomous vehicles operating in challenging environments, Int. J. Robot. Res. **25**(5–6), 449–483 (2006)
22.68 P. Bellutta, R. Manduchi, L. Matthies, K. Owens, A. Rankin: Terrain Perception for Demo III, Proc. of the 2000 IEEE Intelligent Vehicles Conf. (Dearborn 2000) pp. 326–331
22.69 KARTO: Software for robots on the move. www.kartorobotics.com, accessed Nov 12, 2007 (ISRI, Menlo Park 2007)
22.70 The Stanford Artificial Intelligence Robot: www.cs.stanford.edu/group/stair, accessed Nov 12, 2007 (Stanford Univ., Stanford 2007)
22.71 Perception for Humanoid Robots. www.ri.cmu.edu/projects/project_595.html, accessed Nov 12, 2007 (Carnegie Melon Univ., Pittsburgh 2007)