Pulli, Kari, Vision Methods for an Autonomous Machine Based on Range Imaging
Department of Electrical Engineering, University of Oulu, 90570 Oulu, Finland
Acta Univ. Oul. C 72, 1993
Oulu, Finland
(Received 5 May 1993)

Abstract

Range imagery produces the kind of geometric information about the environment that an autonomous machine needs for operation. Not only does the range data provide explicit information about the obstacles that might hinder movements, but it also provides a means of recognizing objects that should be manipulated.

TULKO, the "Machine of the Future" project, has chosen laser range imagery as it main source of vision data. In TULKO the location and orientation of the work space is determined with respect to a paper roll manipulator. The locations of individual paper rolls must also be determined in such a way that the manipulator can transfer them. In the actual application the manipulator can autonomously load the rolls from a platform to a ship.

This thesis presents a Hough transform based method for locating standing paper rolls of a known radius directly from the depth data. The locations are transferred into the manipulator coordinate system by calibrating the sensor and manipulator coordinator systems.

For object recognition from depth maps, a new method for range image segmentation was created. The method first calculates local surface normals from the depth data using robust methods, and decomposes the normal vector into three orthogonal components. Based on two of the components it is possible to determine discontinuities in the surface orientation. A third component, a scaled depth value, is added to the two normal components, and the resulting triplets are considered as a single color vector with three "color" bands. This multi-band image is then segmented using a hierarchical connected component method. The resulting segmentation is refined by robustly fitting surface equations to the segments and thereby determining at the region borders to which segment the individual pixels belong.

Keywords: machine vision, range image, object recognition, intelligent robots, segmentation

PREFACE

This thesis was written for the degree of Licentiate in Technology in Information Engineering for the University of Oulu.

The laboratory where I worked was the Computer Laboratory, headed by Prof. Pietikäinen and later by Assoc. Prof. Silvén, within the department of Electrical Engineering. The project was a joint project between the University of Oulu, the Technical Research Centre of Finland (VTT), and several private companies.

I would like to express my gratitude to my supervisors Profs. Pietikäinen and Silvén. I am also indebted to the numerous coworkers from both the VTT and the University of Oulu for their help and advice on the many problems I had, both theoretical and practical. Thank you. Thanks are also due to Gordon Roberts, who checked my English. I am grateful to the TEKES and the Tauno Tönning Foundation who provided financial assistance for the project and for my personal research.

Finally, I would like to thank my fiancée Anne for the mental support she has given me while completing this thesis.

Oulu, May 1^{st}, 1993.

Kari Pulli

LIST OF SYMBOLS

1D, 2D, 3D	One-, two-, three-dimensional
CAD	Computer aided design
CCD	Charge coupled device
DOF	Degrees of freedom
GHT	Generalized Hough transform
HT	Hough transform
LFF	Local-feature-focus
LMedS	Least median squared fitting
LSQ	Least-squares fitting
LTS	Least trimmed sum fitting
mW	milliwatts
nm	nanometers
NP	Nondeterministic polynomial time
NRCC	National Research Council Canada
PEM	Planning—Executing—Monitoring
RAG	Region adjacency graph
TULKO	Abreviation for "The Machine of the Future" (Finnish)
VTT	Abr. for the Technical Research Centre of Finland (Finnish)

CONTENTS

1. INTRODUCTION

Autonomous machines are turning from science fiction into real science. Modern machines do more and more by themselves, needing less and less supervision by an operator. Machines are able to do repetitive chores without getting tired. They can complete difficult and dangerous tasks, such as the repairing of nuclear reactors or a space station, without endangering human lives.

For a machine to be called autonomous, it has to be able to cope with its tasks without human intervention. Usually, it must observe its environment in order to understand what other factors apart from itself affect the surroundings. Sometimes, in very controlled surroundings, such as an assembly line of a car factory, where one *knows* where each object is located, the observation of new things can be reduced to minimum, but in a changing environment one needs to sense the changes. Vision is man's most important sense — it has a very wide bandwidth in conveying information, and it allows one to inspect things without touching them. It seems therefore natural that autonomous machines should also be endowed with the ability to *see*.

Although humans see by sensing photons and registering the intensities and the variations of colors in the light, it is by no means self-evident that artificial beings, robots, should also see using the same method. In many robotic applications, it is very important to deduce the geometric or spatial structure of the environment, and the locations and orientations (i.e. the poses) of various objects. The robot may need to be able to recognize objects based on their form, manipulate them, or just move around without bouncing onto walls or other obstacles. The human brain can deduce three-dimensional relations of the objects and entities a person sees, but the process of matching the images of two eyes and deducing range by stereo vision, or using depth cues such as perspective distortion, are very complex. If in most cases what is really needed is geometrical information, would it not be better to obtain this knowledge directly, without intervening processes?

Methods have been developed for directly obtaining range information that actually present the continuous geometry of the environment in a digitized form. In addition to deducing range information from intensity information, it can be produced, for example, by laser ranging devices. *Ladar*, laser radar, is a technique that sends short laser pulses to the environment and measures the range to an obstacle by measuring the time of flight of the light pulse. Many other techniques exist in addition to the ladar.

The TULKO project, or the "Machine of the Future" project, aims to develop a concept for an autonomous machine. Not only concepts, but the project also produces a prototype of an autonomous paper roll manipulator which uses laser ranging as its main means of observing the environment. The project began in 1989, and it ran in three phases. The first phase of the project contained the definition of the problem to be solved, and a survey of methods for solving the problem. Tests were performed by simulating a robot on a computer. The second phase, the laboratory phase, performed tests using real hardware, such as an industrial robot and a laboratory range detector. In the third phase, a prototype of a functioning paper roll manipulator utilizing a laser pointer range detector was constructed.

This thesis concentrates on the object recognition problems encountered in robotic vision when range information is used. During the research, a quick, Hough transform based method for locating paper rolls was devised. Standing paper rolls of a known radius are detected and located using sparse range data from a range pointer. The range device and the paper roll manipulator coordinates are calibrated so that the results can be transferred to the manipulator.

A new method for segmenting range images was also created. The segmentation is based on calculating local surface normal vectors and a scene is segmented into continuous and homogeneous regions that correspond to a natural division of objects into surface patches. The inclusion of direct range information into the process makes the segmentation more robust. Surface equations can be easily fitted into the regions, and ultimately objects can be recognized on the basis of the surface segments and their interrelationships.

The structure of this thesis is divided as follows: chapter 2 presents the TULKO project and the range based imaging as the basis for its vision system; chapter 3 studies the problem of object recognition in three-dimensional vision; chapter 4 presents our method for locating the paper rolls from range data; chapter 5 explains the new segmentation method for range images; and finally chapter 6 contains a discussion and the conclusions of the thesis.

2. A VISION SYSTEM FOR AN AUTONOMOUS MACHINE

Vision is our most powerful, and most complicated sense. It provides us with information about our surroundings and enables us to interact with our environment in an intelligent manner. Although full vision systems for robots are still out of our reach, many important tasks of a vision system can be realized with today's technology. In this chapter, we first consider the goals of a machine vision system. We also take a general look at the TULKO paper roll manipulating autonomous machine, and then a closer look at the vision system.

2.1. Tasks of a vision system

A vision system for an autonomous machine can have several tasks that it should attend to. Some of the tasks, such as navigation and obstacle avoidance, are needed only for a mobile machine, whereas all kinds of robots need to be able to recognize and locate objects.

Obstacle sensing. All the autonomous machines we are interested in have moving actuators or they can move as a whole. In such a case, the robot should be able to determine if it is about to run into some obstacle. Once the danger of collision is detected, the robot can react by either halting or by modifying its trajectory. A blind robot can easily damage not only itself but also objects and persons in its vicinity.

Object recognition. For the machine to know how to react to the objects it senses, the objects must be recognized. The machine must be able to distinquish a person walking in the working area from the paper rolls it is supposed to load into a container for obvious safety reasons. The word recognition implies that something is already known about the object: a machine cannot recognize an object for which it does not have an internal model.

Object pose. Autonomous machines operate on objects by moving, assembling, etc., which usually requires direct contact with the object. The object's pose, i.e. its location and orientation, are important pieces of information when planning how it should be gripped or pushed.

Environment Modeling. For a machine to be able make plans it has to have some kind of model of its environment. If an environment model has not been given beforehand, the machine has to create it by itself by sensing and locating, possibly even recognizing, objects.

Navigation. Mobile autonomous machines must be able to navigate within their surroundings. In navigation the environment model, or map, must be updated and the robot needs to deduce its own position in relation to the map by observing and locating beacons.

2.2. TULKO

2.2.1. An autonomous paper roll manipulator

In 1989 the TULKO project began. This is a joint project between the University of Oulu and the Technical Centre of Finland, and it will continue until May 1993 (Pieskä *et al.* 1991). The name TULKO is an abbreviation of the project's Finnish name "Tulevaisuuden Kone" or "The Machine of the Future", and it is supported by the Technology Development Centre of Finland (TEKES) and by several private companies.

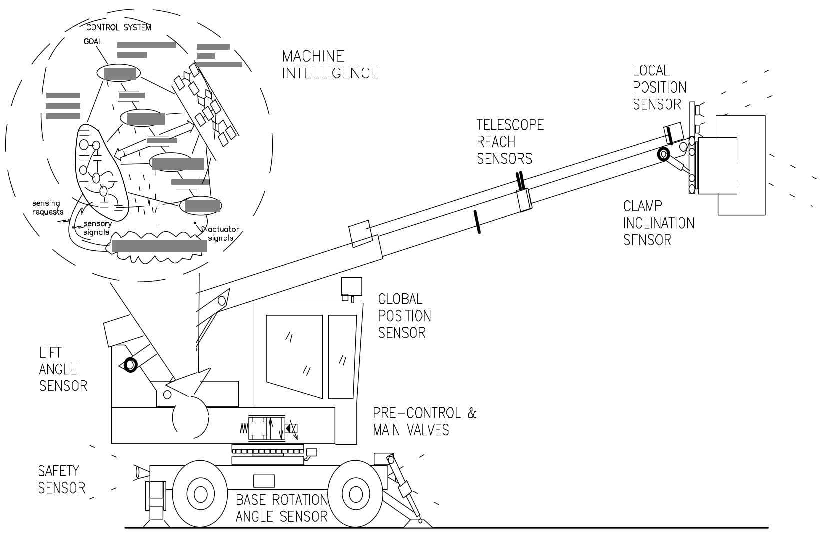


Fig. 2.1 The TULKO manipulator and its main parts, including the control system.

The main goal of the research in TULKO is to increase the autonomy of machines by developing intelligent control systems and sensors. The control system is part of a larger system, which includes a high-level goal-oriented planner (Riekki *et al.* 1991). We have applied the integrated control system to pick-and-place guidance, utilizing structured light ranging. Other sensors used include force sensors and sonars.

As an instance of an autonomous machine we have implemented a prototype of a paper roll manipulator (see Fig. 2.1). Its intended task is the loading and unloading of paper rolls onto and from ships, railroad wagons, etc. The basic ideas of the control method were first tested with a simulator of an indoor mobile robot (Röning *et al.* 1990). In the second phase, experiments were performed with industrial robots equipped with range and force sensors (Riekki *et al.* 1991). Having concluded the laboratory experiments we expanded our research to an outdoor application where the test equipment includes a large paper roll manipulator equipped with a sophisticated gripping device and the appropriate controls.

2.2.2. System overview

The main task of the manipulator system is to move paper rolls of a known radius from area A to area B in a changing and partially unknown environment. To reach this goal, the system must be able to locate the robot and its gripper, collect data about its environment, combine collected information with what is previously known, and pick up, transport, and put down a paper roll.

2.2.2.1. PEM-model

The control scheme is based on a hierarchically organized set of Planning-Executing-Monitoring (PEM) cycles (Heikkilä & Röning 1992). Every PEM cycle is a goal-oriented module, which consists of three generic activities—planning, executing, and monitoring—and a separate meta control mechanism, which takes care of the control of generic activities inside a PEM cycle. This is illustrated in Fig. 2.2.

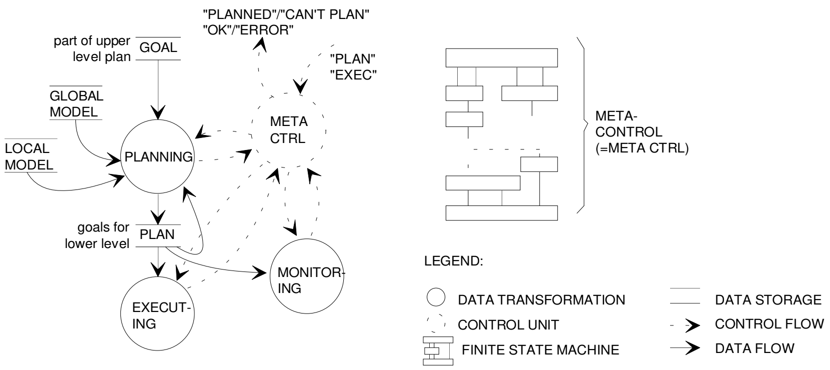


Fig. 2.2 The basic components of the PEM-model.

The *planning* activity encompasses task decomposition by hierarchical resource or activity allocation, and produces plans that the *executing* activity then carries out. While the plan is being executed, the *monitoring* activity monitors the system and its environment, and it initiates replanning if it notices a deviation from the original plan.

The PEM-triplets can be nested at several levels of the hierarchy: an executor may contain in itself several PEM-triplets. This enables the abstraction of a problem so that the higher levels plan the task on a very abstract level, and the lower levels progressively refine that plan.

2.2.2.2. Reactive system

In a traditional robot system, the control system includes all the intelligence required to move a robot arm. It has to notice all the deviations from the plan and modify the plan each time, which slows down the real-time operation of the system. In the TULKO, the goal-oriented planner does not have to take everything into account since some of the intelligence is distributed to the actuators, as shown in the Fig. 2.3. Below the planner there is a reactive system that can meet the goals given by the planner and its role is to adapt locally to unexpected situations.

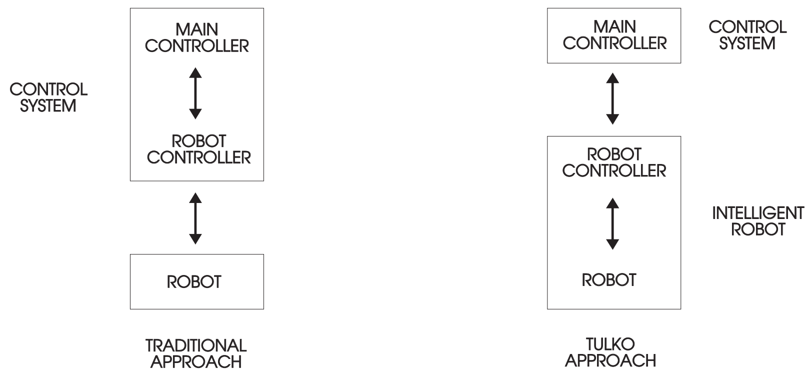


Fig. 2.3 Control of traditional and intelligent robots.

The reactive control is much simpler than the planner. As it is designed for a particular actuator, it can better utilize the available resources. More importantly, however, the reactive control can be made to function very quickly. For example, if a robot is ramming its gripper through a concrete wall, it should be possible to halt it without replanning. Or, if an obstacle is noticed to be in the way of a planned trajectory of the gripper, the reactive system could deflect the gripper's trajectory so that the obstacle can be avoided.

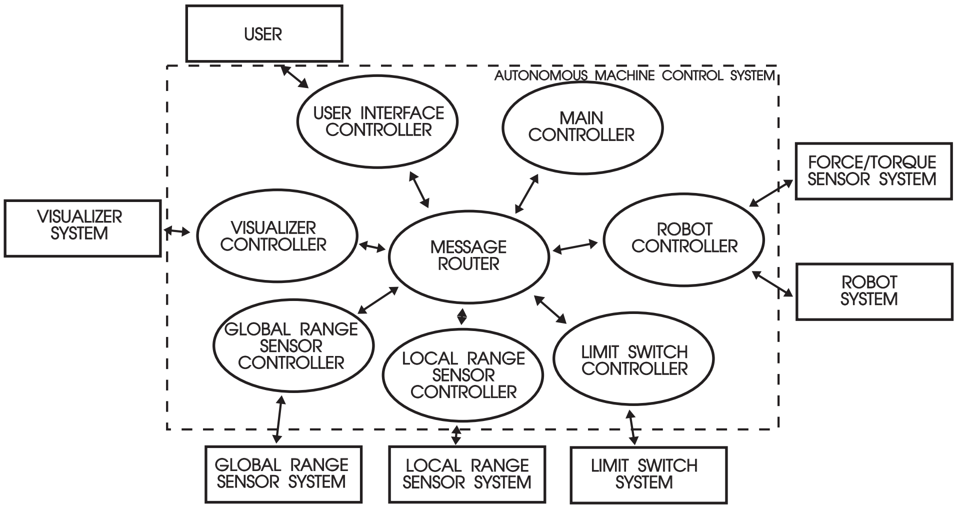


Fig. 2.4 Autonomous machine control system and device systems.

2.2.2.3. System components

The control system is distributed as shown in Fig. 2.4, and consists of the following components: the main controller, the user interface controller, the visualizer controller, the robot controller, the local range sensor controller, the global range sensor controller, and the visualizer system. The components are loosely connected by a message router. This makes the whole system easier to develop and maintain because individual elements can be designed, tested, or changed separately.

The main controller includes the higher levels of the control hierarchy, i.e. the goal-oriented planner, while the low level controls are located in the robot controller. The user interface controller conveys the user's commands to the system while the visualizer system provides a graphical representation of the system and its status. The global range sensor, a laser pointer, is used to position the robot and obtain coarse information about its environment. The local range sensor, a matrix of ultrasonic sensors, provides more accurate information about the position and orientation of selected objects close to the gripper. The force/torque sensor is used to detect and to stop on contact with objects, to verify the success of gripping or releasing an object, and to weigh objects. The limit switches form a safety system: a set of ultrasonic sensors monitors the working area and slows down the system or halts its operation if humans are detected to be too close to the manipulator.

2.3. Global sensor

The global sensor acts as the eyes of the manipulator system. Its tasks include the positioning of the manipulator in respect to the partially known surroundings. To do this, the global sensor should have some knowledge about the geometry of the possible beacons, such as walls, corners etc., sense them, deduce their locations, and from this information infer its own position within the working area. Next, the global sensor locates, using *a priori* knowledge, the areas to transfer objects from and to in respect to its own coordinate system. Last, but most importantly, the global sensor must, while the whole system is operating, locate the objects to be transferred by matching sensor data to inner models.

The global sensor chosen for the TULKO is a laser-based range finding device. In this section, we first examine the characteristics of the range data, and compare them with intensity data obtained from video cameras. The section concludes with a study of different kinds of global sensor hardware both used and planned for use within the TULKO project.

2.3.1. Geometrical information

In three-dimensional (3D) vision geometrical information about objects is needed to recognize those objects and especially to determine their location and orientation. Range data is especially well suited for obtaining such geometrical information.

In the last decade, there has been a proliferation of techniques for producing range images or depth maps and for analyzing them (Besl & Jain 1985; Besl 1988a). Range data is often represented in the form of a matrix of numbers, where the numbers quantify the distances to object surfaces along rays emanating from the sensor focus point and passing through points on a regularly spaced grid. These range data matrices can be used directly as approximations of the 3D shape of the object surfaces within the field of view.

Although the geometry of the objects in a scene is one of the main contributor to the intensity images obtained from video cameras, there are many other variables that affect the result. The type of lighting of the scene, whether it is directed or diffuse, has a great effect on the appearance of an image. Neighboring objects may cast shadows or reflect light on other objects, and surface markings hamper the image segmentation process. Different materials have different reflectance characteristics. Humans are able to compensate for the lack of depth information by the use of higher level spatial reasoning and inference processes, but those processes, along with the associated databases, are very difficult to implement for deducing surface geometry from intensity images. Range data can be used more quantitatively for object geometry reconstruction than the intensity data because of the explicit shape information in range images.

Range imagery has yet another advantage over intensity images in robot vision. Since range measuring devices measure the distance to the closest object surface in each particular direction, it is possible to deduce whether certain locations are free or occupied. Elfes (Elfes 1990) has produced occupancy grids, which explicitly maintain probabilistic estimates of the occupancy state of each cell in a spatial lattice. The information on occupancy can be used in a variety of robotic tasks such as obstacle avoidance, map making, and multi-sensor integration, where the occupance grid provides a common representation for different kinds of sensor data.

2.3.2. Global sensor hardware

Range imaging systems can be divided into two categories: passive and active systems. *Passive systems* use an intensity image produced by a video camera, while *active systems* send and receive a signal, often structured light or ultrasound.

The most important passive range imaging system is the binocular stereo, which uses two cameras and deduces distances using the disparities between the camera images. Other systems can be referred to as the shape-from-X techniques, where the X can be replaced by one of the following list: motion, shading, texture, and contours. Shirai (1987) gives a good presentation of passive range imaging techniques.

There are at least six different optical principles that have been used to actively obtain range images: radar, triangulation, moire, holographic interferometry, focusing, and diffraction. Besl (1988b) has carried out an excellent survey of these methods. Among the active methods, radar and triangulation can provide the range and resolution acceptable for the purposes of the TULKO global sensors. Indeed, both principles have also been applied. In the following, we briefly describe the range finding equipment we have used.

2.3.2.1. Laser scanner

Ideally the global sensor in the TULKO is a laser scanner situated on top of the manipulator as shown in Fig. 2.1. However, the laser ranging system we had is suitable only for laboratory use. Here we describe our laboratory ranging system. We also used range data from a data library (Rioux & Cournoyer 1989), and we briefly describe the ranging system used for obtaining the images in that library.

The scanner that was used in the laboratory phase is the *Technical Arts 100 A* or the *White Scanner* (Technical Arts 1989). The White Scanner sends a laser beam turned to a plane of light into the scene and calculates distances using triangulation. Its field of depth is 150-300 cm, and the measuring volume at a distance of 225 cm is 80 cm wide and 100 cm high (Riekki 1993).

The parts and the operation principle are presented in Fig. 2.5. The system consists of a light source (10 mW Helium-Neon laser, wavelength 633 nm), three mirrors, a video camera, a computing unit, a user interface, and a connection to a workstation. The light source emits a light ray which is turned to a plane of light using an oscillating mirror. The computing unit calculates points on object surfaces in the scene using the video camera image, as well as knowledge about the orientation and location of the light plane and the video camera.

The operation principle is shown in the left part of Fig. 2.5. If a surface is moved within the scene, there is a corresponding translation of bright pixels in the video camera image. The translation in the horizontal axis of the video image is proportional to the translation of the surface in the z-axis in the sensor coordinates. The x- and y-coordinates can be computed using the light plane orientation, z-coordinate, and the vertical position of a pixel on the screen. Hence, each pixel on the screen maps unambiguously to a (x, y, z) point in the sensor coordinate system.

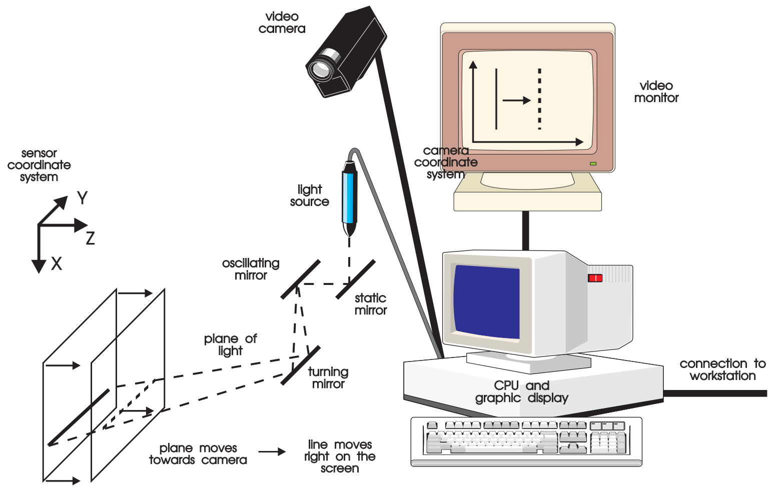


Fig. 2.5 The White Scanner system components and operation.

240 horizontal measurements can be performed at once, and the vertical scanning is implemented by re-orienting the light plane by turning one of the mirrors.

Figure 2.6 shows the optical arrangement of the NRCC ranging device (Rioux & Cournoyer 1989). The device is based on the concept for active triangulation in which the horizontal position detector and the beam projector are both scanned. A turning double-sided mirror produces synchronized projection and detection. As shown in the Fig. 2.6, the beam leaves the source, hits the rotated mirror, and bounces off another mirror and impinges on an object surface. The illuminated spot is viewed via the opposite side of the mirror (and another fixed mirror) by a CCD camera. The CCD array is tilted in order to compensate for defocusing that happens along the z-axis.

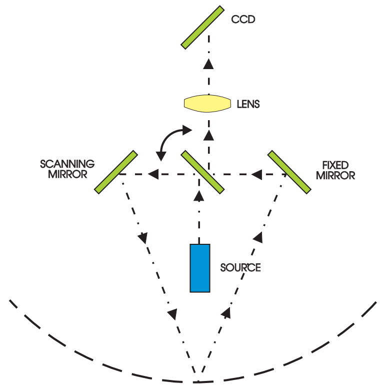


Fig. 2.6 3D measurements using triangulation through synchronized scanners.

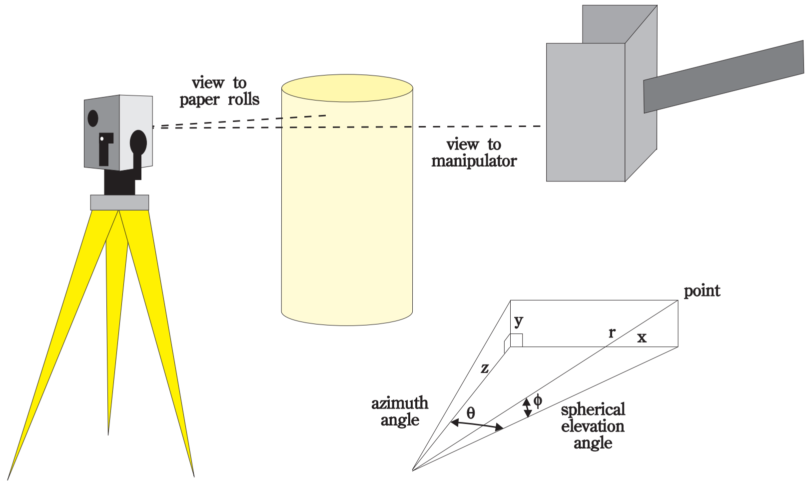


Fig. 2.7 The laser pointer in its operating environment, the returned information.

This system yields a high angular resolution with a small baseline. A 256×256 range image is created in about a second. For a total working volume of 250 mm $\times$ 250 mm $\times$ 100 mm, the x, y, and z resolutions are 1, 2, and 0.4 mm, respectively.

2.3.2.2. Laser pointer

In the final demonstration version of the TULKO, a manually operated laser pointer developed at VTT was used. The user aims the VTT pointer at the scene, presses a trigger, and turns the pointer back and forth over the scene. The device measures single points at regular time intervals. Each measurement consists of the following information: the x-, y-, and z-coordinates, spherical elevation angle ϕ, azimuth angle θ, and distance r. The pointer rests on a tripod where it can see the working area of the paper roll manipulator and the manipulator itself (see Fig. 2.7).

The operation principle of the laser pointer is time-of-flight measurement. Several light pulses are sent, their reflections measured, and the results are averaged. The measuring range of the laser pointer is from 1.5 meters to 10 meters in all directions at a resolution of 1 cm. The measuring accuracy varies from 0.5 cm to 2 cm.

The laser pointer does not produce a depth map, but a number of independent point measurements. Hence, many depth image analysis methods cannot be used with this kind of data.

3. OBJECT RECOGNITION

3.1. Introduction

The ultimate goal of robotic vision is the same as the role of vision for humans: understanding scenes and the spatial relations between the objects within the scene. This involves partitioning the scene into meaningful entities, recognizing individual objects in 3D, and determining the location and orientation of those objects. These results can be used when forming a cohesive interpretation of the visible surroundings. Higher level processes combine the newly deduced information into the beliefs the vision system already has about the structure of the environment, and having recognized objects, make deductions about their purpose and about their potential effect on the robot's own functions.

The word recognition implies that something is already known about the object. If there is no previous knowledge about the object, it can only be described in terms of volume, height, etc., or possibly with vague notions like "roughly spherical" or "cylinder-like", but it cannot be recognized. Hence, an object recognizing system needs to have some kind of models about the objects it knows. These models should, in 3D vision, allow object recognition from an arbitrary view point, i.e. the model should be *view independent*. These object models are stored in a *world model*, a necessary component in a object recognition system.

How should one begin to interprete the abundant digitized sensor data? A brute force method to determine the presence of objects would be to transfer all possible combinations of all known objects in any orientation into the digitized sensor data format and try to minimize the matching error. Clearly, even for very simple scenes, this would take an enormous amount of processing time. We need to have a better way of proceeding.

A much better approach is to reduce the large dimensionality of the input data by transforming it into some kind of symbolic form. If the object models can also be described in this common intermediate domain, subsets of data can be matched against individual objects or parts of objects. The quantities used for matching in the intermediate domain are called *features*.

In this chapter, we first present the components for a recognition system, along with the mathematical description for object recognition and the characteristics a recognition system should have. We also take a look at the object models and intermediate representation possibilities. Segmentation is a natural way of reaching the intermediate representation from the sensor data. Having achieved this, the object matching can continue in the intermediate domain. We then continue with the presentation of the Hough transform, a powerful tool for object recognition. Although the Hough transform can be directly used to recognize objects, it is more efficient to first organize the input data and then to provide the Hough transform with the resulting structured information. The generalized Hough transform is shown to implement object-feature graph matching efficiently.

3.2. Recognition system

3.2.1. Recognition system components

The logical components of a complete object recognition system and their interactions can be depicted as in Fig. 3.1 (Besl & Jain 1985). The four fundamental system domains are: the real world domain, the digitized sensor data domain, the symbolic description domain, and the modeling domain. Several processes map information from one domain to another. The image formation process (I) creates intensity or range data based purely on physical principles. The description process (D) extracts relevant application-independent features from the sensor data. This part should not use any *a priori* information about the particular objects likely to be seen in the image but be totally data-driven, i.e. only information about the image formation process and some basic information about real-world geometrics should be included. The modeling process (M) provides the system with models of real-world objects. Modeling can be totally automated with the use of sensors and methods for organizing sensor data, or more commonly, the models are formed by people, using CAD-systems, for example. The recognition or understanding process (U) matches the symbolic descriptions or features with the object models. Finally, the rendering process (R) can be used for verification. The rendering process produces synthetic data from object models and enables us to find out how closely the current scene interpretation corresponds to the actual measured data.

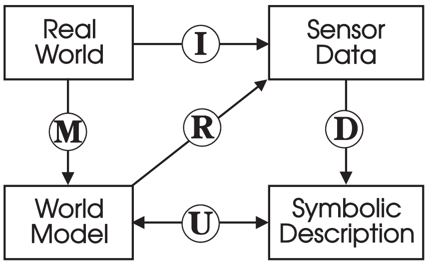


Fig. 3.1 General object recognition system structure. The processes are: I, image formation; M, world-modeling; D, description; U, understanding; R, model rendering.

3.2.2. *Characteristics of a recognition system*

An object recognition system, on the whole, has to meet several criteria. Besl (1988a) has proposed a list of characteristics that an ideal object recognition system should have:

- The system must be able to handle data from an arbitrary viewing direction.
- The system must handle arbitrarily complicated real-world objects.
- Arbitrary combinations of objects should be handled without being sensitive to minor occlusions.
- A certain amount of noise should be tolerated.
- The scenes must be analyzed quickly and correctly.
- The system should be able to express its confidence of its interpretation of sensor data.
- The system should be able to analyze, model, and describe new objects.

The last requirement includes a learning capability, which is a difficult research issue in its own right.

3.2.3. *Mathematical description*

Object recognition from depth maps can be defined as generalized inverse set mapping (Besl 1988a). The world can be approximated to consist of N_{obj} objects, the i^{th} object being denoted A_i. Each object has its own coordinate system so that the origin lies at the center of its mass and the three orthogonal axes are aligned with the principal axes of the object. There is also a world coordinate system which is used to describe the spatial relationships between each object and the rest of the world. Each object can be located within the world coordinate system by means of six parameters: three for translation ($\alpha = (\alpha, \beta, \gamma)$) and three for rotation ($\theta = (\theta, \phi, \psi)$). The coordinate systems and their parameters are illustrated in Fig. 3.2 (a).

The *world model* W is now defined as a set of ordered triplets

$$W = \{(A_i, \alpha_i, \theta_i)\}, \quad 0 \leq i \leq N_{obj}, \tag{3..1}$$

where A_0 is the sensor at α_0 and orientated to θ_0. A time-varying object model can be represented as having the elements of each triplet as a function of time. The set of all objects, the *object list*, is denoted as $L = \{A_i\}$. The sets of translations and rotations are denoted as $\Re^t$ and $\Re^r$, respectively, where $\Re$ is the set of all real numbers and $t = r = 3$. Now the depth map projection of a scene can be modeled

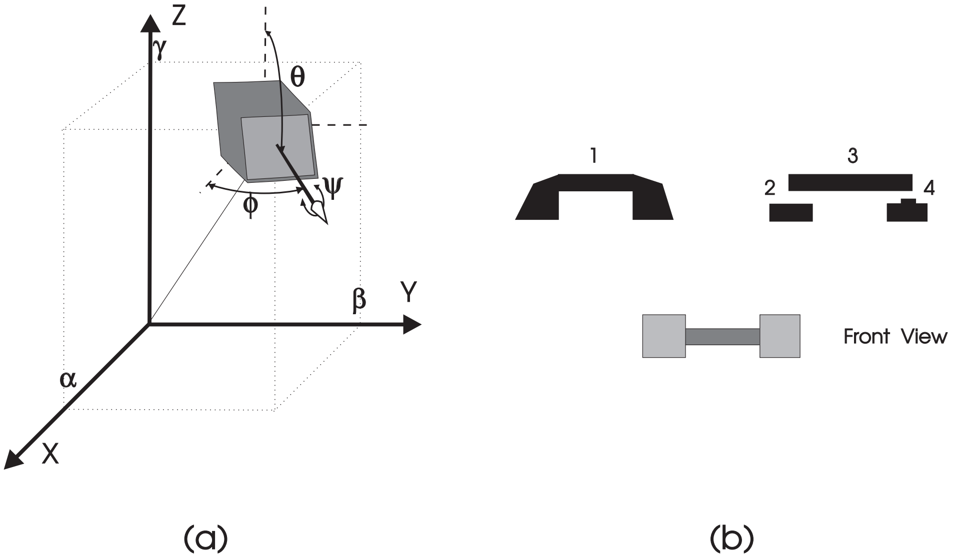


Fig. 3.2 (a) Rigid objects in 3D have 6 degrees of freedom: translation (α, β, γ) and orientation (θ, ϕ, ψ). (b) Different valid interpretations of a simple scene: $\mathbf{P^{-1}} = \{\{\mathbf{1}\}, \{\mathbf{2}, \mathbf{3}, \mathbf{4}\}, \{\mathbf{4}, \mathbf{3}, \mathbf{2}\}\}$.

as a mathematical operator P, which maps elements in the set $\Omega = L \times \Re^t \times \Re^r$ into elements in the set of all scalar functions F of $t-1$ variables:

$$P : \Omega \to F. \tag{3..2}$$

The projection P uses sensor location α_0 and orientation θ_0 as implicit arguments and it produces a function $f(\boldsymbol{x})$, where $\boldsymbol{x}$ is the vector of $t-1$ spatial variables of the sensor's focal plane. The value of the $f(\boldsymbol{x})$ is the distance to the object surface, and in the case, where the point $(\boldsymbol{x}, f(\boldsymbol{x}))$ cannot lie on an object surface, we assign $f(\boldsymbol{x}) = \infty$. Therefore, in the presence of multiple objects, we define $f(\boldsymbol{x})$

$$f(\boldsymbol{x}) = \min_{1 \le i \le M} P(A_i, \alpha_i, \theta_i). \tag{3..3}$$

The depth-map object recognition problem can be now stated: given a depth map function $f(\boldsymbol{x})$, determine all the possible sets of objects and corresponding translation and orientation parameters that could have caused such a projected function. This corresponds to inverse mapping

$$P^{-1}(f(\boldsymbol{x})) = \{\omega_j \subseteq 2^{\Omega} \mid \min_{j \in J} P(A_j, \alpha_j, \theta_j) = f(\boldsymbol{x})\}. \tag{3..4}$$

Each ω_j is a set of objects that projects the $f(\boldsymbol{x})$. However, the $P^{-1}(\cdot)$ is a one-to-many mapping, as several different sets of objects can produce the same $f(\boldsymbol{x})$ (see Fig. 3.2 (b)). The inverse mapping takes elements of the depth-map function space into the power set of the power set of Ω:

$$P^{-1} : F \to 2^{2^{\Omega}}. \tag{3..5}$$

2^{2^Ω} denotes all the possible sets of all the possible combinations of all objects.

3.2.4. Models of objects

What kinds of models do we need within an object recognition system? Fan (1990) has given a set of criteria for 3D object shape description:

- **View-point invariance.** The description should enable recognition from an arbitrary viewing direction.
- **Richness.** The object description should contain enough information for similar objects to be recognized. It should also be possible to recreate reasonably similar objects from that description.
- **Stability.** Minor local changes caused by noise, slight deformations, and digitization errors should not radically alter the description.
- **Local support.** Real scenes always contain occlusion, but objects should be recognized even if they are only partly visible.
- **Naturalness.** The description should correspond to physical features.
- **Reliability.** The error in computation must be always reasonably small.
- **Efficiency.** One must be able to perform the computation within a reasonable space and time.

Unfortunately, some of the requirements are mutually exclusive: the richer the description, the more costly it is manipulate.

Objects can be described at different levels, with low level descriptions such as a set of points or pixels, at intermediate levels such as edges and contours, or with high level descriptions such as surface descriptions or even volume descriptions. With lower level descriptions the dimensionality of the description is very high, i.e. a large number of parameters is needed for the object description, and they are not very stable in respect to viewing directions. Further, they are often sensitive to occlusion. Higher level descriptions have lower dimensionality and they maintain their invariance in different surroundings, but the algorithms for manipulating them are often very expensive or cumbersome. For example, volume descriptions are of a very high level, but currently techniques for computing them exist mostly for simple and regular shapes.

In general, one should choose the description at as high a level as possible while still maintaining robustness of description and efficiency of manipulation. From previous methods, surface description is a good compromise: it is much richer than point-wise or edge-based descriptions, but it is still fairly easy to work with. There is still the question what kind of surface descriptions to use. *Global surface descriptions* describing the whole of the object surface at a time are not very practical: in the case of complex objects they tend not be stable, furthermore, they tend to be very sensitive to occlusion. *Segmented surface descriptions*, on the other hand, describe parts of an object, and the whole object description consists of a set of surfaces and their mutual relationships. This approach is less sensitive to occlusion. One question still remains: how is the surface segmentation to be achieved? One possibility is to approximate the surface by planar or simple curved patches. If the approximation is not good enough, the segmentation is refined until the fitting error falls below a preset threshold. This method, however, is rather unstable, as the description is likely to change considerably even if the described surface changes only slightly. The number of patches is often quite large, and the points and lines where the approximating patches are joined do not necessarily correspond to physically prominent features. A better choice is to have surface patches that are segmented along physically significant features, such as surface orientation discontinuities. If the sensor data can be similarly segmented, the matching of the data against the inner models will be greatly eased.

3.3. Segmentation

An *image segmentation* is the partitioning of an image into a set of non-overlapping regions whose union is the entire image (Haralick & Shapiro 1992). The purpose of image segmentation is to decompose the image into parts that are meaningful with respect to a particular application. For example, in two-dimensional (2D) part recognition, a segmentation might be performed to separate 2D objects from the background as in Fig. 3.3. In 3D object recognition, the segmentation may help in matching sensor data against object surface patch descriptions.

Segmentation is used to simplify the visual input to the level that is required for the specific task. In robotic vision, to simplify means to partition images into entities that correspond to individual regions, objects, and parts in the real world and to describe these entities only in sufficient detail for performing a required task (Bajcsy *et al.* 1990). Thus, the segmentation can be seen as a compression of abundant measurement data into a symbolic form to facilitate higher-level scene

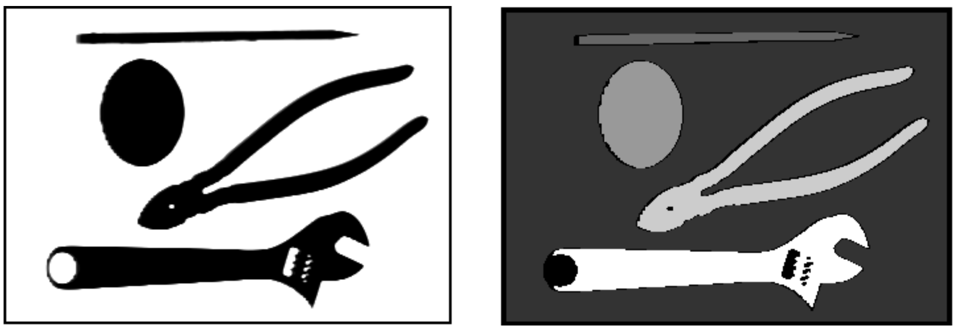


Fig. 3.3 Tools lying on a table have been first separated from the background and then the different regions have been labeled.

analysis processes.

Although it is in general difficult to say what constitutes a meaningful segmentation, some basic rules exist (Haralick & Shapiro 1992):

- Regions of a segmented image should be uniform and homogeneous with respect to some characteristics, such as gray level in intensity images or surface continuity in range images.
- Region interiors should be simple and without many small holes.
- Adjacent regions of a segmentation should have significantly different values with respect to the characteristic on which they are uniform (e.g., they belong to different surfaces or objects).
- Boundaries of each segment should be simple, not ragged, and must be spatially accurate.

It is often difficult, if not impossible, to achieve all the stated properties. Strictly uniform and homogeneous regions tend to be full of small holes and have ragged boundaries. Insisting that adjacent regions have large differences in values may cause regions to merge and boundaries to be lost.

3.3.1. Mathematical description

The general segmentation problem can be mathematically stated as follows (Horowitz & Pavlidis 1974; Zucker 1976; Besl 1988a): Given the set of all image pixels I and a logical uniformity predicate $P(\cdot)$, find a segmentation S of the image I in terms of a set of regions R_i. The following segmentation conditions must hold for the set S:

$$\begin{gathered}\bigcup_{i=1}^{N_R} R_i = I, \quad \text{where } R_i \subseteq I \text{ for each } i \\ R_i \cap R_j = \emptyset = \text{Null Set} \quad \text{for all } i \neq j \\ R_i \text{ is a 4-connected set of pixels} \\ \text{Uniformity predicate } P(R_i) = \text{TRUE for all } i \\ \text{If } R_i \text{ adjacent to } R_j \Longrightarrow P(R_i \cup R_j) = \text{FALSE}.\end{gathered} \tag{3..6}$$

N_R is the number of regions in the segmentation. The result of the segmentation process is the list of regions denoted

$$S = \{R_i\}, \quad 1 \leq i \leq N_R, \tag{3..7}$$

which is usually accompanied by an adjacent regions list S_A where the pair (i, j) indicates the adjacency of the ith region and the jth region:

$$(i, j) \in S_A \Longleftrightarrow R_i \text{ adjacent to } R_j. \tag{3..8}$$

The pair (S, S_A) is known as the region adjacency graph (RAG), where the regions are the nodes of the graph and the adjacency relationships are the arcs of the graph. A more detailed description of RAG's will be given in 5.5.2..

3.4. Hough transform

Hough transform (Hough 1962) is a robust method for detecting patterns in images by transforming image features into points in parameter space and locating clusters thus formed. Good sources for Hough transform are the survey of Illingworth and Kittler (1988), and Davies (1990), which presents Hough transform as the most important intermediate level machine vision method.

In this section, we first study the basic Hough transform for line detection in intensity images. We then proceed to the generalized Hough transform, and conclude with methods for improving the effectivity of the Hough transform.

3.4.1. Introduction into Hough transform

The Hough transform (HT) (Hough 1962) was first introduced for detecting curves in bubble chamber photographs. The key ideas of the method can be illustrated by studying an example where the line passing through colinear image points is detected.

Image points (x, y) that lie on a straight line can be described by a relation

$$f((\hat{a}, \hat{b}), (x, y)) = y - \hat{a}x - \hat{b} = 0, \tag{3..9}$$

where the parameters a and b characterize the line (the hats denote that the parameters are kept constant). Equation 3..9 performs a one-to-many mapping from the space of parameter values to the space of image points. The HT's main idea is to invert the mapping of Eq. 3..9 and *backproject* an image point to the *parameter space*, i.e. the set of possible parameter values, resulting in a many-to-one mapping

$$g((\hat{x}, \hat{y}), (a, b)) = \hat{y} - a\hat{x} - b = 0. \tag{3..10}$$

Figures 3.4 (a) and (b) illustrate the backprojection: each image point (x, y) of a straight line backprojects a straight line in the parameter space (a, b). These lines intersect at a common point, the coordinates of which characterize the straight line connecting the image points.

In order to represent the continuous parameter space on a digital computer the parameter space is usually tesselated into a set of regular finite-sized regions. In the previous line detection example, the parameter space can be represented as a two-dimensional array (see Fig. 3.4 (c)). This array is called the *accumulator array* and its elements are called the *parameter cells.* The size of the cells is chosen to correspond to the desired precision of parameter estimation. Each image point casts *votes* that are then accumulated into cells when a backprojected line passes through the parameter space region associated with the element. These votes peak in the cell where the backprojected lines intercept. Hence, the task of finding the interception points can be transformed to the easier task of detecting local peaks in the number of accumulated votes.

The HT can be easily extended to detect other parametrically defined curves. A curve of n parameters

$$f((\hat{a_1}, \ldots, \hat{a_n}), (x, y)) = 0 \tag{3..11}$$

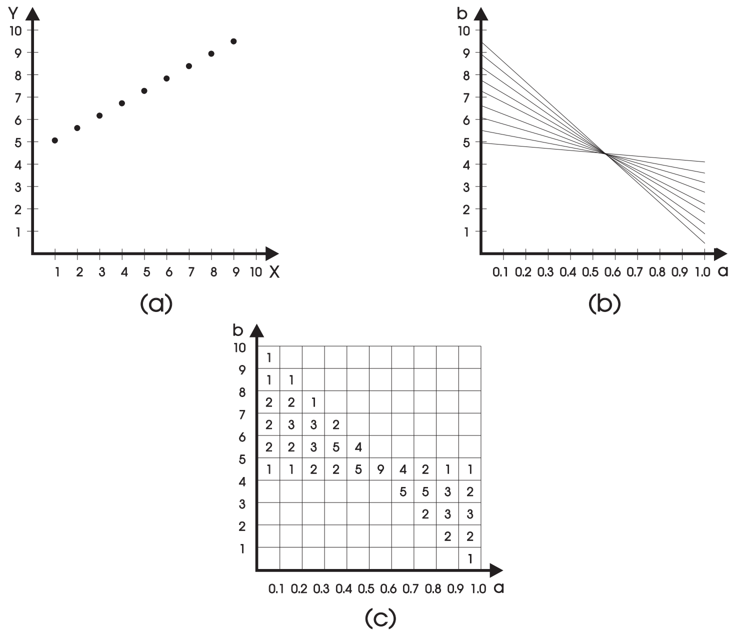


Fig. 3.4 The basis of the Hough transform for line detection: (a) $(\mathbf{x}, \mathbf{y})$ point image space; (b) $(\mathbf{a}, \mathbf{b})$ parameter space; (c) $(\mathbf{a}, \mathbf{b})$ accumulator space.

can be regarded as the backpropagation equation

$$g((\hat{x}, \hat{y}), (a_1, \ldots, a_n)) = 0. \tag{3..12}$$

This equation maps out a $(n-1)$-dimensional hypersurface in the n-dimensional parameter space. Again, the intersection point of those hypersurfaces determine the most probable parameters for image curves.

The Hough transform can be generalized to detect arbitrary shapes for any orientation and scale (Merlin & Farber 1975; Ballard 1981). In the following description we assume that edge points and edge directions have been detected using some edge detectors (c.f. (Canny 1986)).

The generalized Hough transform (GHT) begins with selecting a localization point L within a template of the idealized shape. From each point on the edge we move a variable distance R, in a variable direction φ, so as to arrive at L. Notice that the parameter space is congruent now with the image space. Both R and φ are then functions of the local edge normal direction θ (see Fig. 3.5). With this arrangement, the votes will peak at the preselected localization point L. The functions $R(\theta)$ and $\varphi(\theta)$ can be stored either analytically, or, for totally arbitrary shapes, they can be stored as a look-up table. However, shapes with concavities and holes require $R(\theta)$ and $\varphi(\theta)$ to have multiple values for some values of of θ (Fig. 3.6).

What the HT really does is gather evidence: for each image point, it is assumed that it belongs to the sought-after shape, and a vote is cast for each parameter

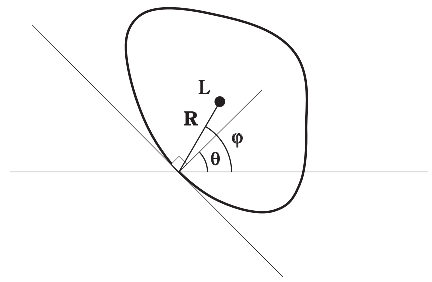


Fig. 3.5 Computation of the generalized Hough transform.

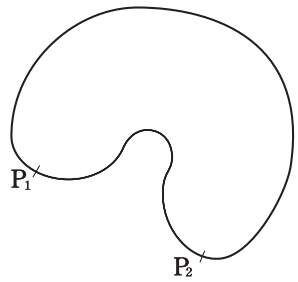


Fig. 3.6 A shape with a concavity: points $\mathbf{P_1}$ and $\mathbf{P_2}$ have the same θ.

combination that includes that point as part of the shape. The accumulated votes then indicate the relative likelihood of shapes described by parameters within the corresponding parameter cell.

The HT method has many desirable features:

- Each image point is processed individually, which enables parallel processing implementation for real time applications.

- Partial or slightly deformed shapes can be recognized because the evidence is gathered independently. For example, in the case of occlusion the HT degrades gracefully because the visible parts still contribute to the correct parameter values.

- The HT method is robust to the addition of random noisy data. This random noise is more likely to cause a low level of background of cell counts than to concentrate on a single cell. However, structured background noise can more easily cause spurious false peaks in the accumulator array, and some care must be taken to eliminate or identify such situations.

- Several instances of a particular shape occurring in the same image can be detected simultaneously. Each instance produces its own peak in the parameter space.

The main drawback of the basic HT implementation is its large storage and computational requirements. In order to resolve n parameters each divided to α intervals, an accumulator array of α^n cells is needed. This can be much too large, especially as the n grows. The bulk of the computational costs comes from the calculation of the intersections of accumulator cells and the $(n-1)$-dimensional parameter surface, and it grows exponentially with the dimensionality of the problem. Also the effort needed to find the peaks in the parameter space grows with the size of the accumulator array.

3.4.2. Improving Hough transform efficiency

The examples of the HT or the GHT have so far been for 2D intensity images. 2D images have only three degrees of freedom (DOF): two for translation and one for the rotation. The situation changes considerably when 3D objects are considered — they have six DOF: three for translation and three for orientation. Conceptually there is no difference, but the computational and storage requirements grow significantly. Here we study some methods for reducing those requirements and for making HT more effective.

The Hough transform problem solution becomes simpler if the problem can be constrained so that the problem dimensionality becomes lower. In the line detection example such constraint could be the line orientation. If both the point location and the line orientation for a point is known, the backprojection into the parameter space results in a point instead of a line. Thus, the dimensionality of the backprojected hyperplane is lowered from one (line) to zero (point). A method called random sample consensus (RANSAC) (Fischler & Bolles 1981) is similar in spirit to this kind of constraint utilization. The original RANSAC, however, randomly picks n pixels for determining the n parameters of a curve and tests the result against the other data, but the application of RANSAC into the HT is straightforward.

Another possibility for lowering the high dimensionality is to decompose the problem into smaller sets of parameters which can be determined sequentially. For example, circles are usually described by 3 parameters, their center coordinates (a, b) and the circle radius r. This 3D problem can, however, be solved in two stages: the first stage involves a 2D HT for determining the center parameters (a, b), while the second stage solves the radius using a 1D HT. Cylinders, which are described by five parameters (two for orientation, two for (x, y) location with $z = 0$, one for radius), could be decomposed into three simpler stages: orientation detection, followed by location detection, and finally the determination of the radius. Hence, a $O(n^5)$ problem is reduced to $O(n^2)$.

The other source for high resource demand for the HT is the size of the accumulator array, which directly grows with the resolution. Many techniques have been proposed that employ non-uniform or multiple resolutions — they have high resolution only in places where a high density of votes accumulate (Li & Lavin 1986; Illingworth & Kittler 1987). Typically these techniques begin the voting process using a coarse parameter space with uniform resolution. Cells with high counts will then be further inspected at higher resolution. Very high parameter resolutions can be reached with only a few iterations. High precision can also be achieved by combining the HT with a least-squares method: first a coarse solution is obtained using the HT, and then outliers, i.e. pixels or measurements not belonging to the sought-after shape, are discarded, based on the results of the HT. A LSQ fitting is performed for the remaining inliers.

Often potentially very large accumulators are needed, but actually only a small fraction of the cells will receive any votes. In such a case, the storage requirements can be alleviated by using techniques for sparse matrices. Xu *et al.* (1990) have presented a dynamic tree structure for the accumulator array. With this method a high resolution can be achieved without a large memory consumption. Also the maxima searching becomes quicker when a far smaller accumulator array has to be sought.

Intensity images, or dense range maps, often contain a large number of possible edge points which can contribute to the HT. It is usually not necessary to process all the edge points — often a small fraction will produce similar results much more quickly. If random samples are taken and their votes are accumulated, the correct peaks denoting instance parameters of the sought-after shape will be formed and can be detected long before processing all the possible contributors (Fischler & Firschein 1987; Xu *et al.* 1990; Shvaytser & Bergen 1991).

3.5. Matching whole objects

In the previous section we studied the Hough Transform and how it could be applied to recognize and locate rather simple objects. For complex shapes, the basic HT or GHT requires a large amount of computation. The task is made easier by recognizing objects from their features. Suitable features are, for example, holes, corners, segmented surface patches etc.; i.e. any readily localizable subpattern. Somehow these features must be organized so that they can be used for recognizing and locating objects. Methods for accomplishing this form the subject of this section.

We begin by examining a graph-theoretic maximal clique approach to object location using point pattern matching, i.e. the features have no other attributes than their x, y, z coordinates. Then, we proceed to show the equivalence of the GHT and the maximal clique approach, and finally we show how the attributes of more complex features can be included in the recognition process.

3.5.1. Detecting objects by matching graphs

In general, we need at least three unique point features to be able to determine both location and orientation of an object in 3D, although with more matched features we can be surer of correct interpretation. There are several reasons for the direct and complete matching of features proving difficult when analyzing 3D data:

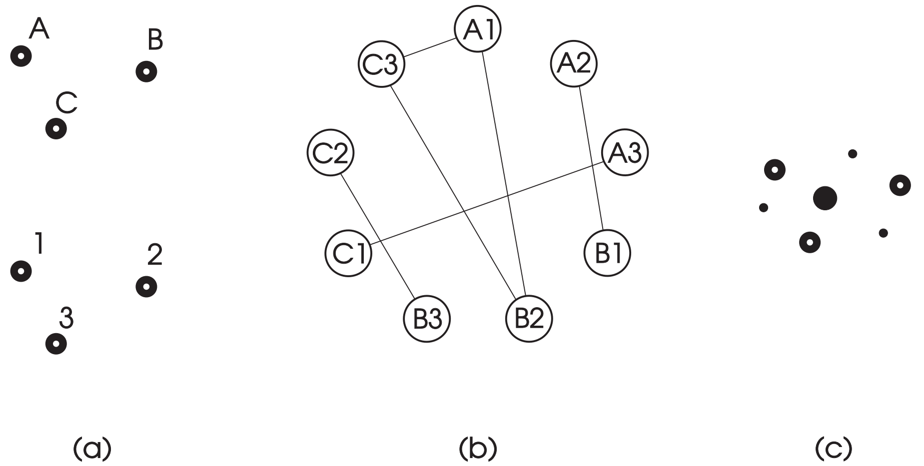


Fig. 3.7 Simple matching of a triangle: (a) basic labeling of the model (top) and the features (bottom); (b) match graph; (c) placement of votes in parameter space.

- Many similar feature points may be present in the data because of multiple instances of similar objects in the data.
- Noise or clutter from irrelevant objects may create additional spurious feature points.
- Certain features of an object are missing because of noise or occlusion.

These problems are best tackled by trying to match as many features as possible so that these matches are consistent with each other. If the matched features are considered to form *graphs* with the matched features as *nodes*, the task can be seen as the mathematical problem of subgraph-subgraph isomorphism, i.e. finding which subgraphs in the *match graph* have a similar structure to a subgraph of the idealized *template graph*. The *arcs* of the match graph represent pairwise compatibilities between matched features.

The match graph can be systematically constructed by numbering the features, labeling the template nodes with letters, and making a node of every possible matching of a detected feature to a feature in the template graph. Arcs will be created between nodes that are consistent with each other, i.e. if node A is a correct match and it is possible that in that case also node B can be a correct match, an arc will connect A and B. A *clique* in a graph is a complete subgraph in the sense that all pairs of nodes are connected by an arc. *Maximal cliques* are the largest internally consistent sets of matched features, and therefore they are the most reliable matches between the data and the object model.

The maximal cliques approach is illustrated in Fig. 3.7. We have found three features 1, 2, and 3, and they are matched against an object model with features

A, B, and C. There are nine possible feature matches, six valid compatibilities, and four maximal cliques, of which the largest one is the correct match.

In general, if a feature is occluded, the number of possible feature matches and valid compatibilities will be reduced, as will also the size of the largest maximal clique, but the maximal clique will still point to the most reliable interpretation. On the other hand, if extraneous features caused by noise or clutter from other objects is present, the number of possible feature matches will grow. The number of the compatibilities will also grow if the extra features happen to appear at permissible locations in comparison to other located features. The size of the maximal clique will remain the same, but the graph will become more tedious to analyze.

As the number of nodes and arcs, and the maximum clique size in the match graph grow, the execution time of the maximal clique problem grows exponentially, i.e. the problem is NP-complete (Aho *et al.* 1974). There are some methods to simplify the problem. One such method is to take advantage of possible symmetries within the object, which may reduce the problem drastically. Another method is called the local-feature-focus (LFF) (Bolles & Cain 1982), which has certain similarities to the RANSAC (Fischler & Bolles 1981). In the LFF one searches for special subsets of features of an object, hypothesizes an object, and verifies the hypothesis against the original image. Because of possible occlusions one may need to search for several different subsets.

The maximal clique approach performs essentially an exhaustive search of fitting matches, and it is effectively a parallel algorithm. This is the source for both its robustness and its slowness.

3.5.2. Object location using GHT

The generalized Hough transform (GHT) is essentially equivalent to the maximal clique approach in that it also performs a complete search. However, the GHT is *not* NP-complete. This is possible because the GHT solves the maximal clique problem in *real space*; it does not solve the *abstract* maximal clique problem.

The GHT can be applied by listing all features and then by accumulating votes in parameter space at each possible localization point, L, consistent with pairs of features (2D problem) or with triplets of features (3D problem). These feature pairs correspond to arcs in the match graph. When using point features, one only needs an R-table indexed by the interfeature distances. Each feature pair gives rise to two votes (or more, if symmetries exist) in the parameter space so that one of the votes is at the correct location. Every peak in the parameter space corresponds to a maximal clique in the match graph: there is a one-to-one relationship between the two concepts. The correct locations all add up to give a large maximal clique and a large parameter peak in the parameter space. This is illustrated in Fig. 3.7 (c).

This basic GHT method is able to locate the objects from their features but not able to determine their orientation. However, it is possible to detect which pixels or measurements belong to the object once the location is known, and then run the HT again to determine the orientation. In the object location, the parameter space is congruent with the image or 3D space; in the orientation detection one can use the Gaussian sphere, where a point on the surface of a unit sphere corresponds to a direction vector. The sphere can be tesselated into triangles of the same area, so that a specific triangle corresponds to two components (ϕ, θ, see Fig. 3.2 (a)) of the orientation vector. The third component (ψ) can be realized with a 1D HT vector within each triangular cell.

If there is more information available about the features than their mere location (e.g., type of the feature, orientation), the maximal clique problem and its GHT solution become easier. The extra information can help to prune some false compatibilities that could not have been detected from inter-feature distances alone. Not only do the extra feature attributes reduce the computation, they also make the interpretation less ambiguous.

3.6. Summary

In this chapter we have addressed the problem of object recognition in computer vision. An ideal system and its characteristics have been described along with the mathematical description of the object recognition problem. The rest of the chapter concentrates on the Symbolic description and World model domains, and especially on the understanding process between them.

The objects are represented by surface patches and graphs that describe the relationships between different patches of the same object. The object model can also include a list of features, e.g., corners or edges, that could facilitate the object recognition.

Segmentation is often used to preprocess the image. Segmentation reduces the huge dimensionality of the input data into a set of homogeneous segments. These segments can then be used as features for recognizing whole objects.

We have presented the Hough transform and the generalized Hough transform, powerful methods for feature extraction and for object recognition. However, if the HT is directly used for recognizing complex objects, the effort needed may become overwhelming. The situation is alleviated by not recognizing objects directly, but by using their prominent features.

The maximal cliques method is a graph theoretical and very robust approach for object recognition using features. Unfortunately, the maximal cliques algorithms are NP-complete. The GHT, however, can be used to solve the maximal cliques problem for the subset of real space representation problems in a polynomial time.

4. LOCATING PAPER ROLLS FOR THE MANIPULATOR

In this chapter, we present the paper roll finding system implemented for the TULKO paper roll manipulator prototype. The system uses sparse range data obtained with the VTT laser pointer range detector. Upright paper rolls of a known radius are quickly and reliably located, and their position is described in the manipulator's coordinate system

The global sensor of the TULKO project, the VTT range detector pointer, was introduced earlier, in section 2.3.2.2.. The pointer produces *sparse range data*. We define the sparse range data to mean range measurements consisting of a set of less than or not much more than 100 range measurements. Furthermore, in our case the pointer is manually operated, and the data it produces is structureless, i.e. it consists of independent (x, y, z) measurements of object surfaces in the environment of the laser pointer.

In the TULKO, the main task of the global detector is to locate the paper rolls for the manipulator to be able to grab them and pile them into, for example, a railway container. It also has to locate some beacons in the partially known environment so that the manipulator's planner knows where to transfer the paper rolls. This is implemented simply by locating the walls with a few measurements and calculating the locations of the corners.

The range sensor has a different coordinate system from the manipulator's coordinate system. Calibration enables locations expressed in one system to be transformed into the other coordinate system.

We first describe our method for locating the paper rolls from the sparse range data, and then we describe how the coordinate systems of the range detector and the paper roll manipulator were calibrated.

4.1. Finding cylinders using the Hough transform

Our task was to locate paper rolls within the working space of the manipulator using 3D range information. The following constraints in the problem make the task easier to accomplish: the paper rolls are rotationally symmetrical, they are assumed to be standing up (their orientation is parallel to the z-axis), and their radius is known. Thus the problem is reduced to a 2D problem: determine the

(x, y) location of an upright cylinder, or if the z-coordinates are ignored, determine the center of a circle with a known radius.

In the laboratory phase of the TULKO project, we used the basic GHT: for each measurement, a vote is cast for all the possible locations of a circle. As we know that valid measurements can be only obtained from one side of the circle or cylinder when using ranging methods, the possible locations in the parameter space, which is congruent to the real space, make a half-circle shaped curve. Figure 4.1 illustrates the voting scheme.

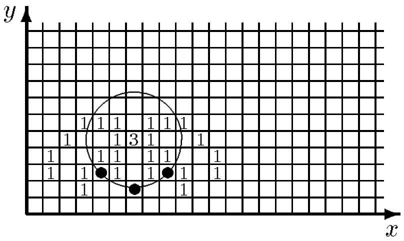
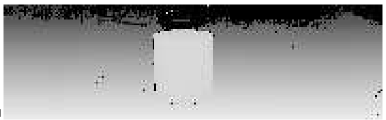


Fig. 4.1 A peak in the Hough accumulator indicates the cylinder's location.

The laboratory phase experiments were performed with the White Scanner range detector (see section 2.3.2.1.). Local tests were made to find out whether a measurement could possibly lie on a vertical surface. Only such points were used to vote for the cylinder location. Figure 4.2 demonstrates how a cylinder is found from the 3D data.

(a)  (b) 

(a) original depth map
(b) points possibly on vertical surface
(c) points close to cylinder center

(c) 

Fig. 4.2 Results from White Scanner data.

As we moved on to implement the prototype of the TULKO paper roll manipulator, we noticed how our scheme could be improved: instead of drawing a half-circle into the parameter space based on single measurements, we can constrain the voting scheme further. In the improved method we vote for the location of a hypothesized circle center based on *a pair* of measurements. Figure 4.3 illustrates how the center of the circle is deduced from a pair of measurements.

Two measurements give rise to two positions for a circle of a known radius. Only a position that is farther away from the sensor than either measurement can be accepted. Hence, the dimension of a hyperplane to be updated in the parameter space is lowered to from 1D to 0D, from a curve to a point. Our method can be

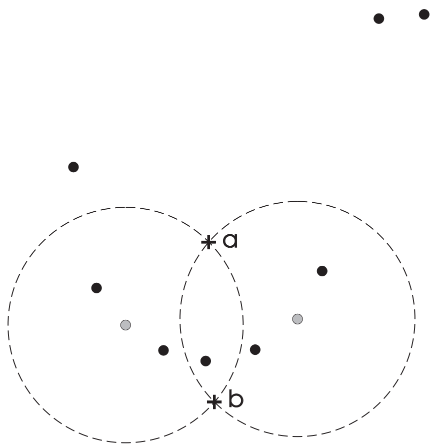


Fig. 4.3 The crosses mark the possible locations of the circle center when only the measurements marked with grey color are considered. The cross b is discarded as it lies closer than the measurements.

easily extended to detecting circles of an unknown radius. Three points lying on a circle uniquely determine all the circle parameters. So, instead of choosing pairs, triplets of measurements are picked up, and the count of the accumulator cell lying on the center of the circle is increased.

The number of possible pairs for n measurements is $\binom{n}{2} \approx n^2$. However, many of these pairs cannot contribute to a vote about the location of a circle simply because their distance is larger than the diameter of the circle, i.e. they cannot possibly both lie on the same circle of a known size. On the other hand, if very close measurements are used, small errors in the measurements will cause large errors in the estimate for the center of the circle. The following algorithm chooses pairs of measurements from the VTT laser pointer data:

```
sort (pointlist, n)
FOR i = 1 TO n-1
  BEGIN
    paircounter = 1
    j = i + 1
    DO
      d = distance (pointlist[i], pointlist[j])
      IF (d > mindist * paircounter) AND (d < maxdist) THEN
        BEGIN
          vote (pointlist[i], pointlist[j])
          paircounter = paircounter + 1
        END
      j = j + 1
    UNTIL (paircounter > limit) OR (j > n)
  END
```

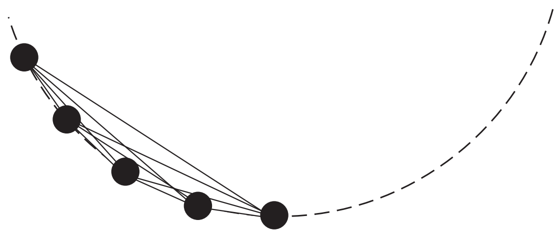


Fig. 4.4 Five points on the circle give ten votes.

First the measurements are sorted from left to right in respect to the field of view of the range pointer. Then each point in the list of measurements is considered, one at a time. A pair is sought in such a way that the points are neither too close nor too far away. When such a point is found, the accumulator is updated based on the point pair. For each point maximally `limit` pairs are considered. The constant `maxdist` states the maximum allowed distance within the point pair, and it is equal to the circle diameter. The `mindist` expresses the minimum allowed distance between the pair of points, so distance between the first pair of points is once the `mindist`, then twice the `mindist` for the second pair, and so on. We have set its value to 1/4 of the circle radius. In our implementation, maximally four points to the right of the current point can be paired with the current point; therefore `limit` is set to four.

Our choice for the values for `mindist` and `limit` permits us to analyze the minimum number of votes that a cylinder, at least half of which is visible, will cast to its center's location in the parameter space. In this case, we assume that at least five points lie evenly spread on the circle (see Fig. 4.4). Out of these five points, we obtain ten pairs of points that will each cast a vote for the location of the circle's center.

In practice, we analyze the measurement data using the previous algorithm. We then search for the cell with the highest number of votes. If the vote count exceeds ten, we deduce the presence of a paper roll. However, there may be more rolls present, and they are searched for next. For each local peak in the accumulator, if the vote count exceeds the threshold 10, *and* the peak's location is not within the circle diameter of a previously found circle center, an instance of a paper roll is deduced.

If the accumulator would be implemented directly as a two dimensional matrix of 16 bit integers, it would require 80 Kbytes of memory for a manipulator work space of 10 m $\times$ 10 m with a resolution of 5 cm. With a resolution of 1 cm the memory requirements would rise to 2 Mbytes. In a single measuring session, only a small fraction of the accumulator cells will receive any votes, and most of the accumulator space is wasted. We have tackled this problem by implementing the accumulator with a dynamic structure similar to the one used by Xu *et al.* (1990).

The dynamic accumulator (depicted in Fig 4.5) consists of a root, a list of x values, and for each x value a list of y values (called a column). The lists are

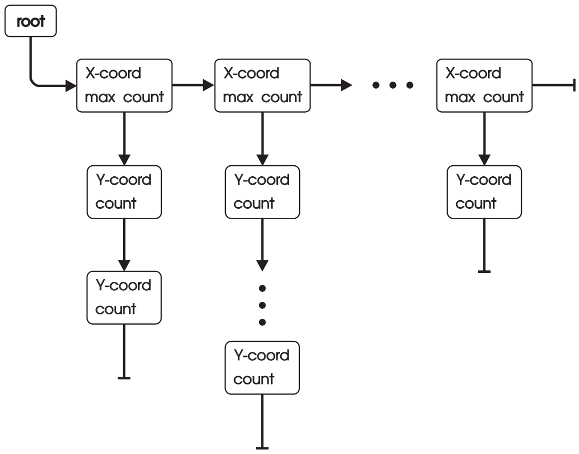


Fig. 4.5 The dynamic accumulator structure.

implemented as one-way linked lists. Each y element contains space for the cell vote count. Each x element contains space for the maximum count for that column. In the beginning, only the root exists and the desired resolution is chosen. For each vote the column and row of the virtual accumulator is deduced. The element is sought by traversing first the x list and then the y list. In the case where the column (x element) does not previously exist, it is created — the same is done with the row (y element) — if it exists, its vote count is increased. At the same time, if the new vote count is greater than the previous maximum count of that row, the counter of the x element is also increased.

On the whole, the space requirements drop dramatically in comparison to the direct implementation of the accumulator. The dynamic accumulator is also unlimited in the sense that it is not necessary to set beforehand any maximum or minimum values for the parameters. The peaks in the parameter space can be found very effectively by first finding the maximum count of the x elements and then the maximum y element in that column. The cost of the dynamic implementation is the greater effort needed for casting a vote to an accumulator cell.

4.2. Calibration of coordinate systems

It is not enough to locate the paper rolls with reference to the laser pointer — the location has to be conveyed to the manipulator in a form which the manipulator understands. The coordinates expressed in the coordinate system of the laser pointer can be transformed to the manipulator coordinate system if the orientation and location of one coordinate system is known with reference to the other.

This transformation can be expressed as follows (Pulli 1991): we define $M_{i \leftarrow j}$ as the transformation matrix which converts a point P_j in coordinate system j to a point P_i in coordinate system i; i.e.

$$P_i \;=\; M_{i \leftarrow j} \cdot P_j. \tag{4..1}$$

The points P are expressed in homogeneous (4D) coordinates, i.e. a fourth component w is added to the x, y, and z components. The transformation matrix $M_{i \leftarrow j}$ can be derived from the position of the origin of the system j (d_x, d_y, d_z) and from the normalized direction vectors for principal axes of the system j (column vectors $\vec{x}_j$, $\vec{y}_j$, and $\vec{z}_j$), all expressed in the system i (homogeneous) coordinates. The axes are direction vectors ($w = 0$), whereas the origin is a point ($w = 1$). Collecting these four column vectors in a 4×4 matrix yields the transformation matrix

$$M_{i \leftarrow j} = \begin{bmatrix} x_{x_j} & x_{y_j} & x_{z_j} & d_x \\ y_{x_j} & y_{y_j} & y_{z_j} & d_y \\ z_{x_j} & z_{y_j} & z_{z_j} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4..2}$$

$M_{i \leftarrow j}$ can be interpreted so that the upper left 3×3 matrix aligns the principal axes of the two systems, after which the rightmost column aligns the origins.

Instead of determining the transformation matrix from the relationships of the coordinate systems, we chose to perform a series of coordinate measurements, so that for each point, the location in both of the coordinate systems is known.

Let $P_j = [X \; Y \; Z \; 1]^T$ be a point in the sensor coordinate system and $P_i = [X' \; Y' \; Z' \; 1]^T$ its coordinates in the manipulator coordinate system. Also, let the matrices A and B be made of several points:

$$A = \begin{bmatrix} X_1 & & X_n \\ Y_1 & \ldots & Y_n \\ Z_1 & & Z_n \\ 1 & & 1 \end{bmatrix}, B = \begin{bmatrix} X'_1 & & X'_n \\ Y'_1 & \ldots & Y'_n \\ Z'_1 & & Z'_n \\ 1 & & 1 \end{bmatrix}. \tag{4..3}$$

Now the simultaneous transformation of several points can be expressed as

$$M_{i \leftarrow j} \cdot A = B. \tag{4..4}$$

If we can find the *pseudoinverse* A^+ of matrix A,

$$M_{i \leftarrow j} = B \cdot A^+ \tag{4..5}$$

is the best approximation for the transformation matrix in the least-squares sense (Lawson & Hanson 1974).

The pseudoinverse A^+ of an arbitrary real matrix A is defined to be the unique real matrix with the following properties:

$$\begin{aligned} AA^+A &= A, \\ A^+AA^+ &= A^+, \\ AA^+ &= (AA^+)^T, \\ A^+A &= (A^+A)^T. \end{aligned} \tag{4..6}$$

(4..7)

Matrix A^+ is reduced to the conventional matrix inverse A^{-1} when A is square and non-singular.

The Ben-Israel-Greville algorithm (Ben-Israel & Greville 1974) is a stable, reliable, and deterministic way of computing the pseudoinverse matrix. The algorithm iterates beginning with

$$X_0 = \alpha A^T, \quad 0 < \alpha < \frac{2}{\mathrm{trace}(AA^T)}, \tag{4..8}$$

where trace is the sum of the diagonal elements of the matrix. The iteration

$$X_{k+1} = X_k(2I - AX_k) \tag{4..9}$$

is performed until $\|X_{k+1} - X_k\| \approx 0$. We have taken the same norm for the ending condition as in (Krishnamurthy & Ziavras 1988), i.e. the absolute value of the maximum element of the difference matrix should be less than 10^{-n}. We chose the $n = 10$, and we break the iteration after 100 iteration if the convergence has not been achieved.

The measurement of the corresponding points has been implemented as follows: a number of locations all over the working area of the robot are chosen. The robot gripper is guided to those locations, and the position of the gripper at each location is measured by manually aiming the laser pointer.

In order to determine the pseudoinverse matrix, we need at least four location pairs. We have tested the system with up to ten point pairs, which seems to be a totally adequate number. No four points should be on the same plane; having four or more points on the same plane makes the matrix A non-singular and effectively reduces the number of useful point pairs in the coordinate system transformation.

We have achieved accuracies in the neighborhood of 20 cm within a working area of about 5×8 meters. The inaccuracies are caused mainly by the fact that the point that should be measured by the pointer lies in the middle of the gripper's wrist joint, and for some gripper positions it is possible to measure its location only with an accuracy of 20-30 cm.

4.3. Summary

An efficient and robust method for locating cylinders of constrained orientation was created. The method is based on the generalized Hough transform concept, and it is able to utilize the structureless range data produced by the VTT laser pointer.

Several improvements over the basic GHT were introduced. The algorithm constrained the accumulator voting scheme by considering several measurements simultaneously. The pair candidates for each measurement were chosen intelligently so that only measurement pairs likely to lie on the same cylinder were considered; this speeds up the algorithm. The potentially large memory requirements were reduced by using a dynamic structure for the voting accumulator. This was important because the whole TULKO manipulator controlling system runs on the same micro computer.

The cylinder locations are found in the sensor's coordinate system, but it is the manipulator that needs to know the cylinder positions. The coordinate system transitions can be carried out using a transformation matrix. By obtaining a set of points, whose coordinates are known in both of the coordinate systems, the transformation matrix can be calculated. The pseudoinverse solution gives the least squares solution for the transformation matrix.

5. SEGMENTATION OF RANGE IMAGES

5.1. Introduction

Range image segmentation differs from intensity image segmentation. In intensity image segmentation the uniformity predicate $P(\cdot)$ (introduced in section 3.3.1.) is usually based on direct intensity values or perhaps textures, i.e. certain local intensity patterns. When range images are segmented based on (depth) value alone, only step edges (see Fig. 5.1), where values suddenly change, can be found. Thus, only objects not touching can be separated. To segment touching surfaces of two objects or different neighboring faces of a single object (roof edges, see Fig. 5.1), one needs to take orientation, which really is a derivative of depth values, into consideration. Hence, the reason why range images cannot in general be segmented using the same algorithms as for intensity images is the uniformity predicate $P(\cdot)$, which needs to consider both the values and their derivatives.

The literature abounds with different approaches to the range image segmentation problem. The two main approaches are region and edge-based segmentation methods. *Region segmentation* attempts to group pixels into surface regions based on some homogeneity or similarity criterion of regions. Some of the methods detect only planar surfaces (Shirai & Suwa 1971; Pong *et al.* 1981; Taylor *et al.* 1989), while others include also quadric surfaces (Faugeras *et al.* 1983; Oshima & Shirai

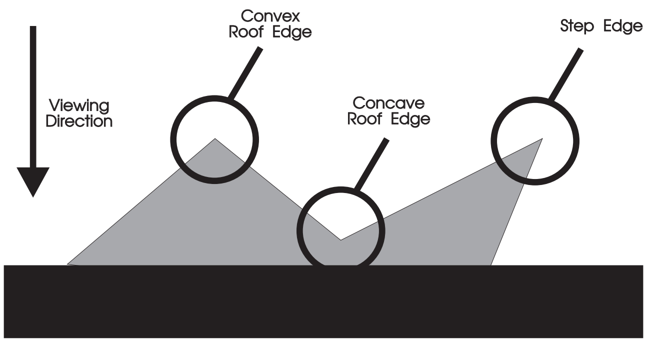


Fig. 5.1 Edge types of interest in range images.

1983). There are also methods that characterize surfaces using tools from differential geometry (Besl & Jain 1988). *Edge detection* tries to extract discontinuities and form a closed boundary around components (Bhanu *et al.* 1986; Tomita & Kanade 1984; Fan *et al.* 1987). Methods exist that combine region segmentation with edge detection, gaining robustness from independent redundant information (Yokoya & Levine 1989; Koivunen & Pietikäinen 1992; Davignon 1992).

Oshima and Shirai (1983) first detect small planar regions, and then combine them into planar surfaces. Small and slender regions with large variance are combined into quadratic surfaces. Besl and Jain (1988) estimate view-invariant Gaussian and mean curvatures of surfaces, and on the basis of their signs, classify surfaces into eight classes. The centers of large regions with similar characteristics are chosen as seed regions, where segmentation by region growing is initiated. Fan *et al.* (1987) calculate the surface curvature and locate step and roof edges, as well as ridges (smooth extrema), using the zero-crossings and extrema of the curvature along several directions. Separate edge points are then combined into a closed boundary. Yokoya and Levine (Yokoya & Levine 1989) use a hybrid of the region and edge-based methods. They combine three methods: the differential geometry approach of Besl, step edges derived from depth values, and roof edges from the partial derivatives of the depth values.

Some methods are mainly based on the homogeneity of surface normals. Dane and Bajcsy (1981) form Gaussian images for segmentation. Jiang and Bunke (1989) operate directly on needle maps and form first surface patches, which are then merged into planar regions, and some of those regions are in the end merged into curved regions. The segmentation method of Taylor *et al.* (1989) is a split-and-merge method, where the homogeneity criterion is based on the comparison of two angles describing the normal orientation and the original range value. Merging is based on a simple minimum and maximum value comparison of neighboring regions. Sabata *et al.* (1990) use the homogeneity of normal vectors and their three projections onto the xy-plane, the yz-plane, and the zx-plane. After the initial clustering the method proceeds to refine the clustering iteratively using a pyramidal algorithm. Four independent segmentations are made to form an oversegmented image. These are then merged by higher-level routines (e.g., variable order bivariate polynomial fitting).

Edge-based methods have to connect points of discontinuity into borders, and these into *closed* region boundaries, which is not a trivial task. Region based methods that are not limited to planar patches, even when aided by information about discontinuities, produce a segmentation only after fitting a surface equation to each resulting region. Clearly, even assuming that explicit surface description is needed for *each region* of an image, that description is easier to obtain *after* the segmentation.

In the rest of the chapter we present a simple but powerful range image segmentation method based on fusing local approximations of normal vectors with depth information (Pulli and Pietikäinen 1993). The result of the segmentation is a set of homogeneous surface regions that need not be planar—they can also be curved. The algorithm converts a depth map to local normal vectors, using their x- and y-components to detect orientation discontinuities, and the depth map to detect

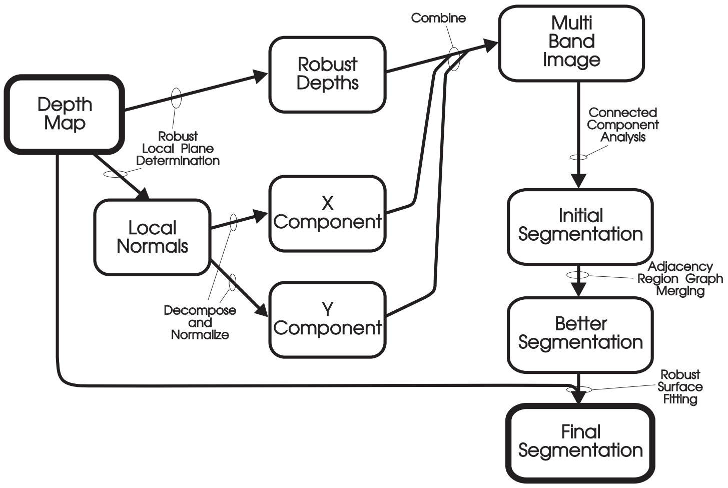


Fig. 5.2 Range image segmentation through decomposition of normal vectors.

depth discontinuities. These three components are treated as three color bands, and the resulting image can be segmented using a region growing type color image segmentation method. We use a method consisting of connected component analysis (Haralick & Shapiro 1992) followed by merging of a region adjacency graph (Zucker 1976) which was developed for color images (Westman *et al.* 1990). Using the results of the connected component analysis, we show that polynomial surface equations describing the surface geometry can be easily found, and they can in turn be used to further refine the earlier segmentation result. Figure 5.2 presents a diagram of the segmentation method.

5.2. Surface normal vectors

5.2.1. Normal extraction

There are several methods for obtaining local surface normals from range data. The basic approach would be to fit a continuous differentiable function to data, and compute its derivatives analytically. If the data is clean enough, computationally more efficient methods suffice, such as the local quadratic surface least squares (LSQ) approach presented in (Besl 1988a). Another possibility is to use the slightly more complex local LSQ planar fitting method of Taylor *et al.* (1989).

The problem in the LSQ methods is that the orientation discontinuities get blurred. Further, if there is noise present in the measurement data, the data must first be filtered, which often blurs images even more. Gaussian filtering (or approximately Gaussian such as binomial filtering (Besl 1988a)) attenuates Gaussian noise well but blurs sharp edges. Median filtering removes shot noise without blurring edges, but it also makes roof edges flat.

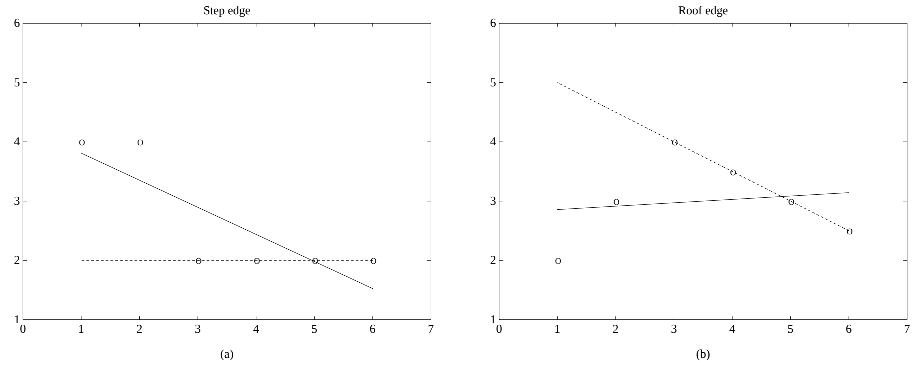


Fig. 5.3 The solid line depicts a least squares fit, while the broken line results from a robust fit. In image (a) there is a step edge, in image (b) a roof edge.

More reliable alternatives to direct LSQ methods and pre-filtering are the so-called robust methods, such as M-estimators (Huber 1981), iterative reweighting least squares (Besl *et al.* 1989), least median squares (LMedS) (Rousseeuw & Leroy 1987) and least trimmed squares (LTS) (Rousseeuw & Leroy 1987; Koivunen & Pietikäinen 1992). Because LSQ methods try to fit a function to *all* of the data, the outliers that deviate a lot from the bulk of the data pull the fit towards them. Robust methods, on the other hand, fit a function to the majority of the data disregarding outliers. Formally this can be described with the *breakdown point*, the smallest percent of outliers that causes incorrect estimates. LSQ methods have a breakdown point at 0%, while Rousseeuw's LMedS and LTS have the breakdown point close to 50%.

The outliers are not necessarily just bad measurements, they often originate from another data population. In the context of plane fitting, this other data population can be measurements from a different object (step edge) or measurements from a different surface of the same object (roof edge). The effect of outliers in an LSQ fit versus a robust fit can be examined in Fig. 5.3, where the robust fit conforms to the majority of the data, while the LSQ fit is perturbed by measurements from a neighboring surface.

We chose to use LTS as our principal method for obtaining normal vectors. It has a high breakdown point, and compared to the LMedS, it has a better convergence rate and a smoother objective function (Rousseeuw & van Zomeren 1990). In LTS we try to find a plane such that the function

$$\sum_{i=1}^{h} (r^2)_{i:n} \tag{5..1}$$

is minimized, where $(r^2)_{1:n} \leq \cdots \leq (r^2)_{n:n}$ are the ordered squares of residuals (a residual is here the distance of the measurement from the plane along the z-axis) and n is the number of measurements. Optimal robustness properties are achieved

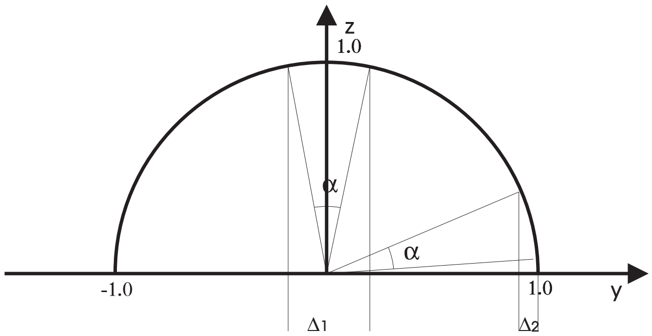


Fig. 5.4 A constant change in the orientation of a normal vector does not map to a constant change in the y-component's value.

when

$$h = \lfloor n/2 \rfloor + \lceil p/2 \rceil, \tag{5..2}$$

where p is the dimension of the function that is fitted (for a plane $p = 3$) (Rousseeuw & Leroy 1987), and $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote rounding to the closest lower or higher integer, respectively. We find the planes used for calculating the residuals (and eventually the surface normals) by choosing, within a local neighborhood, sets of three measurements, and by determining the planes spanned by those point sets. The normal of the plane with the smallest trimmed sum of squared residuals is chosen. The normal vectors are normalized, and if a normal vector points away from the view point, it is flipped.

5.2.2. Normal decomposition

Once we have calculated the local normal vectors, we decompose them into orthogonal x-, y-, and z-components. Later, when we search for homogeneous regions in the image, we do not directly compare the normals, rather these components. Here we notice that the information on the z-component is redundant: since the normal vector has been normalized into unit length, and it has been flipped so that it points towards the viewer, the x- and y-components totally determine the z-component. Therefore, it can be ignored in the later phases.

Taking the x- and y-components apart corresponds physically to having two intensity images of the scene. Both of the images can be thought of as being illuminated by an artificial directed light source, where the light shines from the positive x-axis in one image and from the positive y-axis in the other. The intensity reflected from a Lambertian surface, when lit from the positive x-axis, is the scalar product of the light source direction vector ($\vec{l} = [1, 0, 0]$) with the normal vector. This is equal to the x-component of the normal vector. In our case, we consider both of the images simultaneously, so the situation could also be thought of as having one image with two light sources of different color.

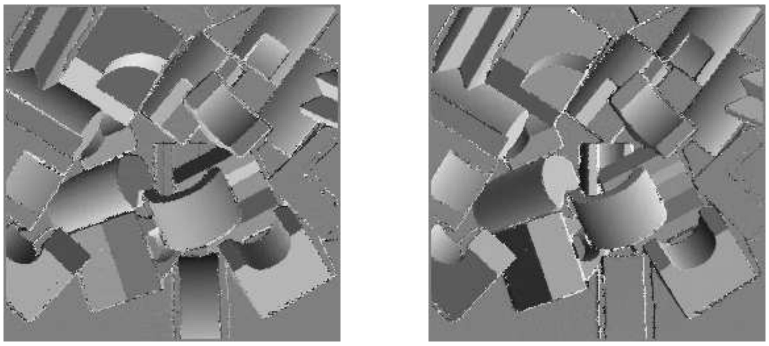


Fig. 5.5 The linearized x- and y-normal components.

We have digressed from the pure physical illumination model on two accounts. First, in our lighting scheme there is no self or other shadowing, i.e. the objects neither cast shadows on their reverse-sides nor over other objects. The negative normal components face away from the light source at the positive infinity, so the corresponding pixels should really be black if the illumination model were to be employed. Instead, the same Lambertian reflection rule that is applied to the surfaces visible to the light is also applied to the surfaces looking away from the light. Another way of looking at this situation is to retain self-occlusion and have another two light sources from the negative x- and y-axes that cast "negative" light.

The second digression lies in the fact that we *linearize* the change of the Lambertian reflectance within curved surfaces. As can be seen in Fig. 5.4, if a normal vector is rotated by a constant angle α, the resulting change in the y-component's value (or similarly in the x-component's value) is not constant ($\Delta_1 \neq \Delta_2$). However, a linear mapping can be obtained by applying the *arcus cosine*-function to each component. This results in a smooth transition of the normal components' values for surfaces with a constant curvature. Figure 5.5 shows images of the linearized x and y normal vector components of a scene of a pile of blocks. The component values are scaled so that normal vectors facing south (with x) and east (with y) will be lighter than vectors facing north or west.

5.3. Depth component

Knowledge about surface orientation is often enough in order to segment images. Take, for example, a black-and-white photograph of an abstract plaster sculpture: a person can usually easily segment that image well, although all the information available is that different surfaces have different levels of brightness, depending on the surface orientation. However, the segmentation process can be aided by the inclusion of depth information. Especially different objects can be easily segmented apart, based on whether they seem to be in the foreground, somewhere in the middle, or in the background. Figure 5.6 shows a scaled depth map.

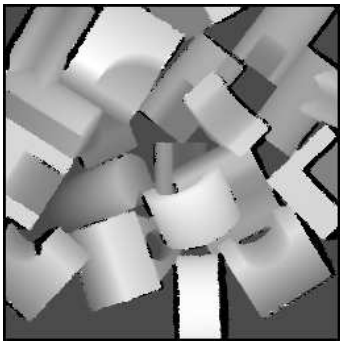


Fig. 5.6 The scaled depth values.

The depth information is directly available from the measured data. However, in order to reduce the impact of random salt-and-pepper noise, i.e. occasional gross errors in depth, we use the robust estimates of local surfaces calculated for determining surface normals: we simply insert the x- and y-coordinates of each measurement to the corresponding plane equation and calculate the z or depth value.

With a view to scaling the depth information and saying what "near" and "far" mean, we need an estimate of the *range* of depth values. Since the original depth data may contain some grossly wrong measurements, and the depth estimates of the previous paragraph may err at some step edges, we do not just take the absolute maximum and minimum depths. Instead, we tesselate the original data into a number of non-overlapping windows, calculate the median depth in each window, and use the maximum and minimum of these medians to approximate the total range of depth values.

5.4. Fusing normal and depth components into a color image

We wish to find homogeneous regions, i.e. regions with no sudden changes of surface orientation or continuity, based on the x- and y-components of normal vectors and distances to surfaces. To avoid the need to have several comparison methods for determining homogeneity, we fuse the different data into a common representation. That common representation is similar to the RGB-model used to represent colors as combinations of three basic colors. Instead of representing red, green, and blue, the "color bands" represent x and y normal components and a depth value.

In the computer representation, each "color band" has the range $[0, 255]$, so the data has to be scaled accordingly. The domain of the x- and y-components is a real scalar $[-1, 1]$, which becomes, after the normalization with the *arc cos*-function, $[0.0°, 180.0°]$. This means that the components' resolution is $180.0°/256 = 0.7°$.

The depth values are scaled using the robust estimates of maximum and minimum distances; in this way we obtain the best contrast. If the scene is very flat, i.e. the maximum and minimum distances do not differ much, we expand the distance

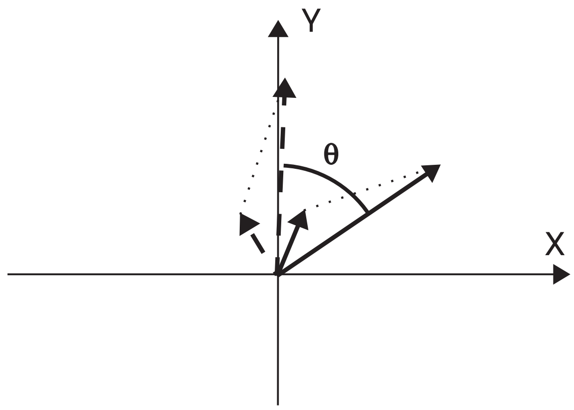


Fig. 5.7 The Euclidean distance between two orientations remain the same when rotated.

span used for scaling in order to prevent too swift a change of contrasts within very inclined surfaces. On the other hand, if the depth values span a large distance, the contrasts may become weak. For this reason the scaling uses also a maximum allowed distance span, which can be obtained from the depth-of-field of the ranging device.

The homogeneity criterion is now the difference, or rather, the similarity, of the "color vector" of neighboring pixels. The Euclidean distance is the correct choice for the metrics of the contrast calculation. Fig. 5.7 shows graphically that the contrasts of normal vector components, when calculated by the Euclidean distance, are rotationally invariant. The solid arrows denote the normalized and combined x- and y-components of two neighboring surfaces. Their difference, or contrast, is marked by the dotted line. The broken arrows denote the surface orientations after the image has been rotated around the z-axis through the angle of θ. The contrast between the two surfaces remains the same. This would not be the case if some other metrics, such as city-block or maximum distances, were to be used.

The process of fusing the information about the surface orientation and distance into a single representation can be viewed as the front-end of the segmentation process. For the back-end, a region growing type color image segmentation algorithm should be preferred, because it tolerates smooth color variations caused by curved surfaces. The hierarchical connected component analysis approach described in the following section seems to be especially suitable for this purpose.

5.5. Segmentation by hierarchical connected component analysis

The segmentation procedure in (Westman *et al.* 1990) is a robust and iterative connected component analysis method for color images, which works by first merging pixels and then regions, depending on their average boundary contrast in the color space. Each successive stage recomputes connected components by a transitive

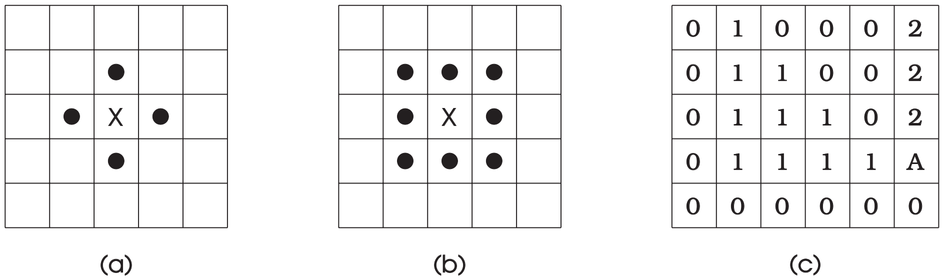


Fig. 5.8 (a) Marked dots are 4-connected to the center pixel x; (b) marked dots are 8-connected to the center pixel x. (c) Pixel connected to two neighbors with different labels.

closure of connectivity among adjacent regions, uniquely identifying the resulting maximal connected components. Usually two stages suffice to produce good segmentation results.

Because the merging criterion is based on the average edge contrasts rather than the maximum and minimum contrasts of the neighboring regions, the method combines into one segment regions where the contrast changes smoothly. This is desirable, as this causes also curved but locally smooth regions to be segmented correctly into a single region. The method results in a reliable and robust segmentation, which can also be efficiently implemented in parallel hardware (Alapuranen & Westman 1992). In the following subsections, we first describe the basic connected component analysis and then the merging of the resulting region adjacency graph (RAG).

5.5.1. Connected component analysis

The first stage computes the initial image segmentation based on the connectivity of the adjacent pixels. The connectivity is determined by calculating the contrasts or the differences in their colors.

Two adjacent pixels p and q are connected if their color difference is lower than a pre-determined threshold value, δ. They belong to the same connected component C if there is a sequence of pixels $(p_0, p_1, \ldots, p_n)$ of C where $p_0 = p$, $p_n = q$, and p_i is a neighbor of p_{i-1} for $i = 1, \ldots n$ (Haralick & Shapiro 1992). Thus, the definition of a connected component depends on the definition of the neighbor. When only the abutting adjacent pixels are considered neighbors, the regions are called *4-connected*, but when pixels with common corners are also considered neighbors, the regions are called *8-connected* (see Fig. 5.8 (a) and (b)).

Let us consider an algorithm for a 4-connected connection analysis. The algorithm makes two passes scanning the image from top to bottom and left to right. In the first pass we examine whether the current pixel is connected to its left or upper neighbor. In such a case, the current pixel receives the same label as the neighbor to which it is connected. If it is connected to neither of them, a new label is created for it. A problem occurs when the current pixel is connected to both neighbors, and the neighbors possess differing labels (see Fig. 5.8 (c)). Then the smaller of the labels is chosen and an entry about the equivalence of the two labels is made to an *equivalence table.*

After the first pass, the equivalence classes are found by taking the transitive closure of the set of equivalences recorded in the equivalence table. Each equivalence class is assigned a unique label, usually the minimum label in the class. Finally, a second pass through the image performs a translation, assigning to each pixel the label of the equivalence class of its first pass label.

5.5.2. Region adjacency graph

After obtaining the basic connected components the procedure is iterated in a second stage, where two components are merged if their average boundary contrast falls below the threshold ϵ ($\epsilon > \delta$). The information that is needed for the merging process, i.e. region labels, average border contrasts, and lengths of the borders between adjacent regions, is stored in a region adjacency graph (RAG).

A RAG can be described as a 4-tuple $G = (V, E, L, \alpha)$, where V is a set of vertices; $E \in V \times V$ is a set of edges; L is a set of labels; and $\alpha : V \cup E \rightarrow L$ is a function that assigns a label to each arc and vertex of G (Zucker 1976). The vertices correspond to regions, and the edges represent the spatial adjacency of these regions, i.e. for two adjacent regions there is a connecting edge. Associated with vertices is regional information, such as area of the region, etc. Edge labels correspond to biregional information, in our case the average border contrast between regions.

The process of region growing can now be modeled in terms of a merging operator defined on these graphs. Let G be a RAG with vertices $i, j \in V_G$ joined by an edge $(i, j) \in E_G$. The merging operator $M(i, j, G)$ constructs a new graph G' similar to G except that the two vertices i and j are joined into a single vertex $i \vee j$ so that all edges (i, k) and $(j, k) \in E_G$ are mapped into $(i \vee j, k) \in E_{G'}$. The vertex $i \vee j$ information is combined from the i's and j's local information, i.e. $\alpha_{i \vee j} = f(\alpha_i, \alpha_j, \alpha_{(i,j)})$; similarly the affected edge labels will be updated, i.e. $\alpha_{(i \vee j, k)} = f(\alpha_i, \alpha_j, \alpha_k, \alpha_{(i,j)}, \alpha_{(i,k)} \alpha_{(j,k)})$. In our case the new vertex information consists of the new label, and the new edge information is the new border length and average border length contrast, which is obtained from old border contrasts and border lengths.

The growth controlling predicate $P(i, j, G)$ is defined to be true whenever $\alpha_{(i,j)} < \epsilon$, where ϵ is the threshold. Now the region growing by a merging algorithm, called the local cluster analysis algorithm, can be written down:

SET $G' = G$
    FOR ALL $(i, j) \in E_{G'}$ DO
        IF $P(i, j, G')$ THEN $G' = M(i, j, G')$.

Figure 5.9 shows the result of connected component analysis and the subsequent region merging for the scene already presented in Figs. 5.5 and 5.6.

5.6. Surface fitting

To be able to describe the regions produced by the connected component analysis, and also refine the segmentation result, we fit surfaces to the measurements within each region. The surface representation we have chosen is a second order bivariate polynomial surface of the form

$$f(x,y) = ax^2 + by^2 + cxy + dx + ey + f = z. \qquad (5..3)$$

We have to use at least the second order if we want to be able to represent curved surfaces. Equations of a higher order, or quadrics, where z is also of the second order, could be better fitted to follow the actual surfaces more closely, but the effort needed to fit them to the data grows with the order. In addition, in higher order representations the surface easily begins to oscillate or undulate.

Earlier, when obtaining the surface normals, we used robust methods in order to reduce the adverse effects of noisy measurements and of measurements belonging to another surface or object. The same policy is followed in the surface fitting. For each large enough region we randomly pick up sets of measurements, and fit surfaces that are close to the measurements in a set. The minimum size of a region is 6 for a second order bivariate surface to be fitted at all, but then locating outliers becomes impossible. Rousseeuw and Leroy (1987) recommend using set sizes that are much larger than $2p$, where p is the dimensionality of the surface (6 in our case)—we use 20 as the minimum size of an area to be fitted. Then, for each surface candidate, all the measurements in a region are compared to the surface equation and the residuals or the fitting errors are computed. Using the residuals we calculate the LTS of Eq. (5..1) and the median of the squared residuals for each surface candidate, and choose the one with the smallest LTS.

Having obtained a good estimate for the surface equation, we fit the surface against all the surface points, except the outliers, using a least-squares fitting. Here, we define outliers as measurements having residuals at least 2.5 times the scale estimate s^0. The scale estimate s^0 corresponds to an estimate for the standard deviation of the data from the surface, and Rousseeuw and Leroy (1987) give a

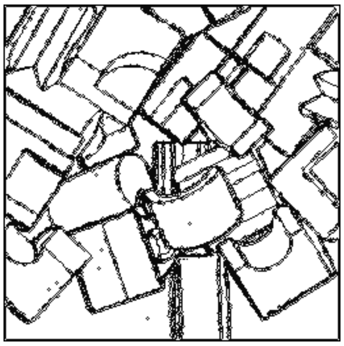


Fig. 5.9 The segmentation result using connected component analysis.

formula for obtaining it:

$$s^0 = 1.4826\left(1 + \frac{5}{n-p}\right)\sqrt{\operatorname*{med}_i r_i^2}. \tag{5..4}$$

The factor 1.4826 is used to make $\mathrm{med}_i\,|z_i|$ a consistent estimate of the standard deviation when the z_i are normally distributed. The second term, which approaches 1 when the sample size n grows, is a correction term for small samples.

For surface fitting, we chose Pratt's (1987) approach. He has presented a direct, i.e. non-iterative and thus fast method for a least-squares algebraic surface fit. First, we have to choose a basis for the surface, and in our case it is $1, x, y, xy, x^2, y^2$, and z. Each point p_i to be fitted forms a row $[1\ x_i\ y_i\ x_iy_i\ x_i^2\ y_i^2\ z_i]$ of the $n \times 7$ matrix A:

$$A = \begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 & x_1^2 & y_1^2 & z_1 \\ 1 & x_2 & y_2 & x_2y_2 & x_2^2 & y_2^2 & z_2 \\ \vdots & & & \vdots & & & \vdots \\ 1 & x_n & y_n & x_ny_n & x_n^2 & y_n^2 & z_n \end{bmatrix}. \tag{5..5}$$

We then have to calculate the 7×7 matrix A^TA and compute its Cholesky decomposition without square roots $U^T D\, U$, where D is a diagonal matrix and U is an upper triangular matrix with diagonal entries equal to 1. The result will be obtained from the U by deleting its last row and inserting the basis polynomials in their place. The determinant of that matrix is the LSQ fitted surface equation.

The A^TA matrix can be computed incrementally: one starts with an empty matrix, and for each point p_i one calculates $A_i^TA_i$, where A_i is the i^{th} point, and add it to the A^TA matrix of the earlier points. Another possibility is to multiply

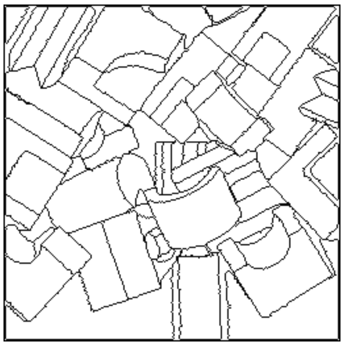


Fig. 5.10 The refined segmentation result using the surface equations.

A^TA out:

$$\begin{bmatrix}
\sum_{i=1}^{n} 1 & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} y_i & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} y_i^2 & \sum_{i=1}^{n} z_i \\
\sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i^2 y_i & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i y_i^2 & \sum_{i=1}^{n} x_i z_i \\
\sum_{i=1}^{n} y_i & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i^2 & \sum_{i=1}^{n} x_i y_i^2 & \sum_{i=1}^{n} x_i^2 y_i & \sum_{i=1}^{n} y_i^3 & \sum_{i=1}^{n} y_i z_i \\
\sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i^2 y_i & \sum_{i=1}^{n} x_i y_i^2 & \sum_{i=1}^{n} x_i^2 y_i^2 & \sum_{i=1}^{n} x_i^3 y_i & \sum_{i=1}^{n} x_i y_i^3 & \sum_{i=1}^{n} x_i y_i z_i \\
\sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 y_i & \sum_{i=1}^{n} x_i^3 y_i & \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^2 y_i^2 & \sum_{i=1}^{n} x_i^2 z_i \\
\sum_{i=1}^{n} y_i^2 & \sum_{i=1}^{n} x_i y_i^2 & \sum_{i=1}^{n} y_i^3 & \sum_{i=1}^{n} x_i y_i^3 & \sum_{i=1}^{n} x_i^2 y_i^2 & \sum_{i=1}^{n} y_i^4 & \sum_{i=1}^{n} y_i^2 z_i \\
\sum_{i=1}^{n} z_i & \sum_{i=1}^{n} x_i z_i & \sum_{i=1}^{n} y_i z_i & \sum_{i=1}^{n} x_i y_i z_i & \sum_{i=1}^{n} x_i^2 z_i & \sum_{i=1}^{n} y_i^2 z_i & \sum_{i=1}^{n} z_i^2
\end{bmatrix} \quad (5..6)$$

and inspect its terms. Here we notice that the A^TA matrix is symmetrical, and some of its terms within the upper part appear more than once, so we only need to calculate a few sums of polynomials, from which the A^TA matrix can be formed.

Our LSQ-approach has a singularity in the coefficient of the last basis term: the last coefficient always equals 1. In the functions of our basis, the z's coefficient should always be non-zero. For this reason, A's columns must be ordered so that z is in the last column.

After the connected component analysis, there are typically some large regions and between them many tiny regions. When we have fitted a surface to the large regions, we can merge the pixels belonging to the small regions by comparing to see if they fit the surface equation of a neighboring large region well. This pixel-wise merging is continued until there are no more pixels left without an associated surface description. Figure 5.10 presents a refined segmentation result which is achieved with the help of the fitted surface equations.

5.7. Results

We have tested our segmentation method on several range images containing both planar and curved objects. The images were obtained from the NRCC (National Research Council of Canada) range image library (Rioux & Cournoyer 1989). The images are of good quality, but we have tested the effect of adding salt-and-pepper noise to the images.

Normal vectors are approximated both by using the LTS method (Rousseeuw & Leroy 1987) and by using a local quadratic surface LSQ method (Besl 1988a). The methods were implemented for 3×3, 5×5, and 7×7 neighborhoods. The LSQ method is a very fast mask-based method. The full implementation of the LTS method, on the other hand, would mean processing through all the possible triangles within a neighborhood.

In order to accelerate the processing, only 5, 10, and 20 preselected triplets for 3×3, 5×5, and 7×7 neighborhoods, respectively, were used. The triplets were not

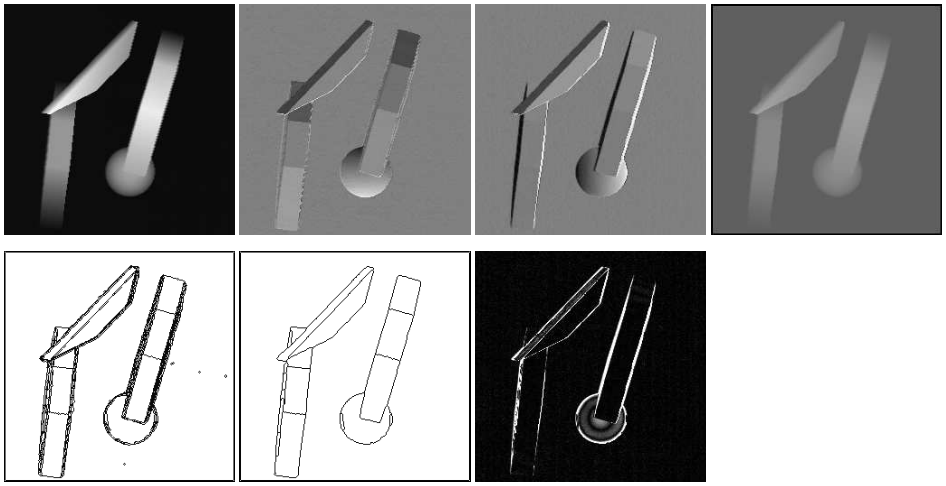


Fig. 5.11 The segmentation of a few blocks using a 5×5 operator size (LTS method). From left to right, top to bottom: original depth map, normal x- and y-components, scaled robust depth map, segmentation before and after surface fitting, and error image.

selected randomly for two reasons: to avoid selecting the same triplets again, and to avoid degenerated triangles that do not span a triangle (i.e. the measurements form a straight line). A third reason is speed: it is faster to consult a look-up table for triangle vertices than to generate them randomly.

When using the robust LTS method for normal extraction, we obtain also a robust estimate for the current range measurement from the plane equation by using the x and y values of the current measurement. With the LSQ method we already have traded robustness for speed and we have no plane equations to use for estimating robust depths. With the LSQ normal extraction we therefore take the measured depths as they are.

We obtain a robust estimate of the dynamic depth range by tesselating the depth image of the image into 5×5 pixel non-overlapping windows, and calculating the median depth in each window. The minimum and maximum values are used for linear scaling of the depth values to $[0, 255]$.

Some examples of the results, using LTS normal extraction, are presented in Figs. 5.11 and 5.12. The images from left to right, top to bottom contain the original range image (with added noise in Fig. 5.12), the linearized x- and y-components of the normals, the depth image, the segmentation results before and after surface fitting, and an error image. In the examples, the first image in the bottom row is the result of the segmentation by connected component analysis. The error image has been calculated from the surfaces that were fitted and from the original measured depths before adding any noise. Dark areas correspond to a good fit, and white means a fitting error of one millimeter or more.

The segmentation result can be seen to be good, although there are many small regions between large, solid segments. Based on this segmentation, we fit surface

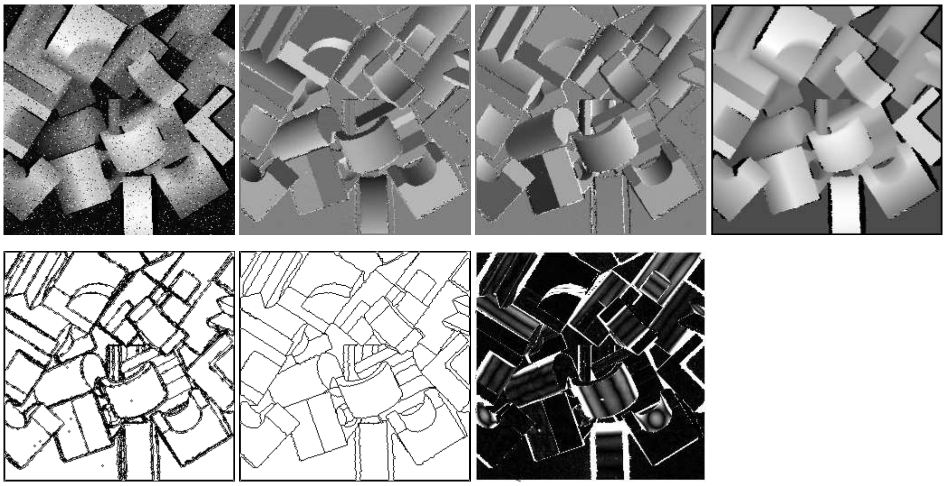


Fig. 5.12 The segmentation of a few blocks using a 5×5 operator size (LTS method). 10 % of the data is contaminated by impulse noise. From left to right, top to bottom: original depth map, normal x- and y-components, scaled robust depth map, segmentation before and after surface fitting, and error image.

equations to the measurements within large segments. These equations are then used to further refine segmentation, the results of which can be seen in the second bottom row image. Fig. 5.13 shows a wireframe image of the noisy data used in Fig. 5.12 along with the resulting surfaces.

The normalization of the normal components was found to be very important; if the x and y components were left unnormalized, curved surfaces were often fragmented into several regions. The resolution of the depth components varies depending on the dynamic range of the depth values in a scene, but we have put limits on the accepted depth range. The upper limit is due to the depth of field or useful operating range of the range finding device, and the lower limit is set in order to avoid too swift change of contrasts within very inclined surfaces. In such a case, the connected component analysis would break that surface into separate segments.

When using the robust LTS method for normal extraction, the segmentation does not seem to be sensitive to the choice of thresholds. Too low thresholds result in several tiny isolated regions, while setting thresholds too high leads to undersegmentation. A safe way is to use smaller thresholds and more merging iterations. There are also ways of choosing thresholds based on image complexity using a cumulative difference histogram (Westman *et al.* 1990). In all the images one basic segmentation and one merging phase were used. We used 4-connectivity, and the thresholds used were 7 (δ) and 15 (ϵ). They correspond to 4.9° and 10.5°, respectively, in the surface orientation differences. In the depth component, the thresholds correspond to, for Fig. 5.11, $\delta = 2.2$ mm and $\epsilon = 4.8$ mm, and in Fig. 5.12, 2.5 mm and 5.3 mm depth differences.

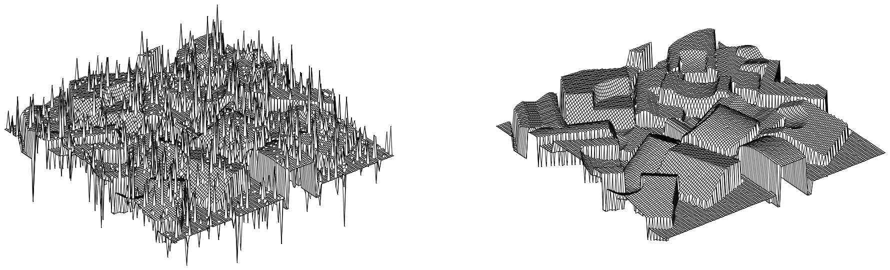


Fig. 5.13 Two wireframe images: the left image shows the noise contaminated data and the right image shows the fitted surfaces.

The LSQ method tends to blur images and smoothen the transition from one surface to another, which makes it more difficult for the connected component analysis part to separate slightly inclined surfaces. This blurring can be observed by thicker region borders (many small regions) between large homogeneous regions in Fig. 5.14, though a smaller operator size (3×3) was employed. If larger operator sizes are used, the blurring effect is also obvious to the eye in the normal component images. Also in the LTS method the normals tend to sway when close to step edges, but the barrier thus formed is narrow and may contain holes, so the depth information is needed to keep the regions apart.

Figure 5.14 demonstrates that a good segmentation result can often be achieved based only on surface orientation information. Still, the depth component should be used, as it adds to the robustness of segmentation, especially in the case of parallel surfaces at different depths.

The LTS method tolerates impulse noise very well, but LSQ based methods do not. On the other hand, LSQ methods tolerate small Gaussian noise better than LTS. In order to get the best of both worlds, one would first have to apply LTS, determine the outliers (very noisy measurements or measurements belonging to another surface), remove them and use some LSQ method to determine the surface orientation. Of course, this would have wide repercussions on the processing costs.

The different parts of our method, at least up until the surface fitting, can be realized quite straightforwardly in real time. The normal estimation for each measurement neighborhood is inherently parallel, and requires only limited local information. The hardware implementation of the LTS method is more involved than that of the LSQ method, but possible. In the LTS implementation, the most demanding part is associated with median calculation, for which hardware implementations exist. Hardware implementation for the hierarchical connected component analysis has been studied recently in our laboratory (Alapuranen & Westman 1992).

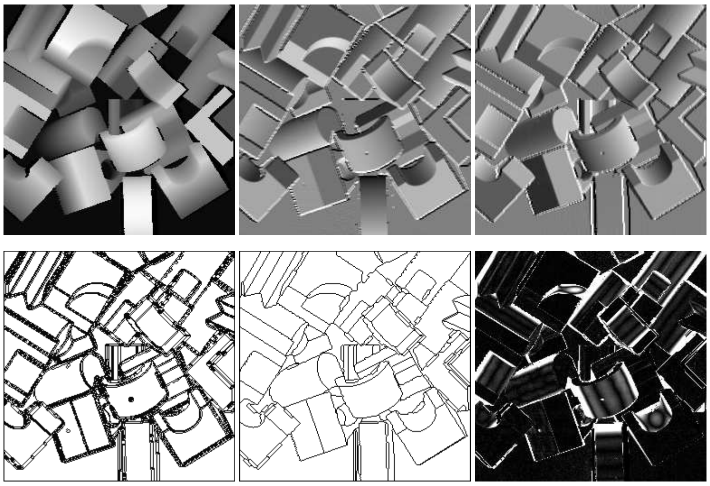


Fig. 5.14 The segmentation of a pile of planar and curved blocks using a 3×3 operator size and the LSQ method. Depth information is not used in segmentation.

5.8. Summary

A simple but powerful method for range image segmentation has been presented. Range images are segmented into homogeneous regions consisting of planar and curved surfaces. The comparison of surface normal vectors is decomposed into the comparison of normal vector x and y components, which are normalized with respect to angular changes. The depth component is also included in the segmentation process.

The results of the decomposition are treated as a three-band color image which is segmented using a hierarchical connected component method. The decisions about merging neighboring regions are based on the average contrast between those regions; this results in a more robust segmentation than if the merging decision was based on the regions' maximal contrast differences. The merging criterion also connects curved surfaces instead of splitting them into planar patches. The normalization of the normal vector components and using the Euclidean distance as the metrics for contrast calculation produces a rotationally invariant and viewpoint independent segmentation result.

The segmentation method presented consists of rather simple components, and all the decisions made need only local information, be it the local depth values or local contrast differences. The processing effort needed is therefore quite constant, and does not depend on the image complexity.

The method yields a robust segmentation without the need for applying resource consuming variable order surface fitting. However, once segmentation has been obtained, surface fitting can be carried out quite easily. The method is suited to scene analysis processes used by, e.g., intelligent robots: it is rather simple and can thus be made to function fast, and it reliably separates different objects.

6. DISCUSSION AND CONCLUSIONS

Image interpretation is a very difficult process. More than 20 years of research have not yet revealed a coherent methodology for vision and image understanding. This is probably due to the nature of the process—also human vision seems to consist of a large set of different processes, competing and co-operating ones, each giving clues that are somehow combined to form an opinion of what is seen. At least the current state of the computer vision research seems to be very much like this: a set of competing and sometimes co-operating methods, the results of which should be drawn together — somehow.

For a robot that is to operate on objects within its environment, it is important to be able to model its surroundings. This should happen even if the scene could not be understood in the sense that the identity and pose of individual objects could be determined. The reason for this is the avoidance of collisions when the robot or other objects move. Therefore, also quite unrefined geometrical knowledge about the surroundings is very important for a moving robot. A range device can acquire this kind of information directly, whereas most depth cues, such as perspective distortion, are available from intensity images only after images are at least partially understood.

In this thesis, we have concentrated on the problems associated with the problems of object recognition. It is very important to have a knowledge base of the objects that the robot is likely to encounter. Without such a knowledge base, object recognition is impossible, as the recognition process is basically a comparison of input data against models of objects and reporting the matches. At the level of individual objects we chose to represent objects with a set of surface patches. The borders of the patches should follow the natural edges of the objects, i.e. lines formed by discontinuities of the surface orientation. There are several methods for representing the individual surface patches; we use second degree polynomial equations mainly because they are simple to operate with computationally, *and* they allow curved surface representation. The object surfaces denote the nodes in the graph that represents the whole object, the arcs of the graph mark the relations between surface patches. In addition to this graph, features useful in object matching, such as holes or prominent corners, can also be added to the object model.

A method that seems to be very general and useful in many different phases of the object recognition process is the Hough transform (HT), especially the generalized

Hough transform (GHT). It has been shown to be equivalent to template matching (Stockman & Agrawala 1977) and also to spatial matched filter (Sklansky 1978). The GHT can locate features from the images by mapping the image data to a parameter space. It is a robust method, that is able to perform well in the presence of noise and occlusions. The GHT can also be used to effectively solve the maximal clique problem in determining which subset of image features is most consistent with the object template graph.

We have used the GHT for detecting paper rolls in an industrial application. An autonomous paper roll manipulator detects standing paper rolls by using range measurements. The measurements are obtained from a manually operated range detector. The range detector could be automated as well: the operator does not have to specifically aim the detector at the rolls, it is sufficient if the scene is panned so that measurements will spread out evenly. Our method locates the paper rolls from this unorganized depth data quickly and reliably. The algorithm is robust and works in the presence of minor occlusions. Furthermore, the space requirements of the algorithm have been reduced so that the whole manipulator control system and the paper roll detection system can both operate on the same micro computer. The paper roll locations are initially known only in respect to the range sensor, but after the manipulator and the range sensor have been calibrated, the locations can be transformed to the manipulator coordinate system.

In a more general system, where we could use a dense depth map, other methods for more general object recognition should be devised. We already know that objects can be recognized by matching surface patches, and their relations, to the models in the vision system's knowledge base using the GHT method. How do we obtain the patches from the depth map? The answer is depth map segmentation by discontinuities in both the depth and surface orientation.

In this thesis, a simple but powerful method for range image segmentation was presented. Range images are segmented into homogeneous regions consisting of planar and curved surfaces. The comparison of surface normal vectors is decomposed into the comparison of normal vector x and y components, which are normalized in respect to angular changes. Also the depth component is included in the segmentation process.

The results of the decomposition are treated as a three-band color image which is segmented using a hierarchical connected component method. The decisions about merging neighboring regions are based on the average contrast between those regions, which results in a more robust segmentation than if the merging decision was based on the regions' maximal contrast differences. The merging criterion also connects curved surfaces instead of splitting them into planar patches. The normalization of the normal vector components and using the Euclidean distance as the metrics for contrast calculation produces a rotationally invariant and viewpoint independent segmentation result.

The method yields a robust segmentation without the need for applying resource consuming variable order surface fitting. However, we have demonstrated that once we have obtained the segmentation, surface fitting can be performed quite easily. The method is suited for scene analysis processes used by, e.g., intelligent robots: it is rather simple and can thus be made to function fast, and it reliably separates different objects. All the decisions made need only local information, be it the local depth values or local contrast differences. The processing effort needed is quite constant and does not depend on the image complexity.

Since our segmentation method concentrates on finding discontinuities in surfaces or in their orientation, it is much better suited to analyzing a scene of multiple objects, than to modeling a single, smoothly varying curved object, such as a person's face. In such a case, one should either use higher order surface descriptions, or try to find segments with a constant curvature type (paraboloids, saddle surfaces, etc.) (Besl & Jain 1988; Koivunen & Pietikäinen 1992).

To implement a general vision system is a formidable task, and so far it has not been done. However, a more limited version suitable for an autonomous robot operating in industrial surroundings is possible, and indeed, has also been often done. For a moving manipulator we propose the use of direct depth information obtained from a laser range device. The depth map can be segmented using the new range image segmentation method presented here, and the resulting surface patches can be matched against the object models using the GHT.

7. REFERENCES

Aho, A.V., Hopcroft, J.E. & Ullman, J.D. (1974) The Design and Analysis of Computer Algorithms. Addison-Wesley, New York, 470 p.

Alapuranen, P. & Westman, T. (1992) Integrated co-processor for connected component analysis. (in Finnish), internal report of Computer Laboratory, Univ. of Oulu, Finland.

Bajcsy R., Solina F. & Gupta A. (1990) Segmentation versus object representation—are they separable?. In: Jain, R.C & Jain A.K. (eds) Analysis and Interpretation of Range Images, Springer Verlag, New York, pp. 207-224.

Ballard, D.H. (1981) Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognition 13: 111-122.

Ben-Israel, A. & Greville, T.N.E. (1974) Generalized inverses: Theory and Applications. Wiley, New York.

Besl, P.J. (1988a) Surfaces in Range Image Understanding. Springer Verlag, New York, p. 339.

Besl, P.J. (1988b) Active, optical range imaging sensors. Machine Vision and Applications 1: 127-152.

Besl, P.J. & Jain, R.C. (1985) Three-dimensional object recognition. Computing Survey 17(1): 75-145.

Besl, P.J. & Jain, R.C. (1988) Segmentation through variable-order surface fitting. IEEE Transactions on Pattern Analysis and Machine Intelligence 10(2): 167-192.

Besl, P.J., Birch, J. & Watson, L. (1989) Robust window operators. Machine Vision and Applications 2: 179-191.

Bhanu, B., Lee, S., Ho, C.C. & Henderson, T. (1986) Range data processing: Representation of surfaces by edges. Proceedings of the 8th International Conference on Pattern Recognition: 236-238.

Bolles, R.C. & Cain, R.A. (1982) Recognizing and locating partially visible objects: The local-feature-focus method. International Journal of Robotics Research 1(3): 57-82.

Canny, J.F. (1986) A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8(6): 679-698.

Dane, C. & Bajcsy, R. (1981) Three-dimensional segmentation using the Gaussian image and spatial information. Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing (PRIP): 54-56.

Davies, E.R. (1990) Machine Vision: Theory, Algorithms, Practicalities. Academic Press, London.

Davignon, A. (1992) Contribution of edges and regions to range image segmentation. Proceedings of the Applications of Artificial Intelligence X: Machine Vision and Robotics SPIE vol. 1708: 228-239.

Elfes, A. (1990) Occupancy grids: A stochastic spatial representation for active robot perception. Proceedings of 6th Conference on Uncertainty and AI, AAAI.

Fan, T-J. (1990) Describing and Recognizing 3-D Objects Using Surface Properties. Springer Verlag, New York, p. 142.

Fan, T-J., Medioni, G. & Nevatia, R. (1987) Segmented descriptions of 3-d surfaces. IEEE Journal of Robotics and Automation RA-3(6): 527-538.

Faugeras, O.D., Hebert, M. & Pauchon, E. (1983) Segmentation of range data into planar and quadratic patches. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8-13.

Fischler, M.A. & Bolles, R.C. (1981) Random sample consensus: A paradigm for model fitting with applications to image analysis and automatic cartography. Communications of the ACM 24(6): 381-395.

Fischler, M.A. & Firschein, O. (1987) Parallel guessing: A strategy for high-speed computation. Pattern Recognition 20: 257-263.

Haralick, R.M. & Shapiro, L.G. (1992) Computer and Robot Vision, vol. 1. Addison-Wesley, New York.

Heikkilä, T. & Röning, J. (1992) PEM-modelling: A framework for designing intelligent robot control. Journal of Robotics and Mechatronics 4(5): 437-444.

Horowitz, S.L. & Pavlidis, T. (1974) Picture segmentation by a directed split-and-merge procedure. Proceedings of the 2nd International Joint Conference on Pattern Recognition, pp. 424-433.

Hough, P.V.C (1962) A method and means for recognizing complex patterns. U.S. Patent 3069654.

Huber, P. (1981) Robust Statistics, John Wiley & Sons, New York.

Illingworth, J. & Kittler, J. (1987) The adaptive Hough transform. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9(5): 690-697.

Illingworth, J. & Kittler, J. (1988) A Survey of the Hough Transform. Computer Vision, Graphics, and Image Processing 44: 87-116.

Jiang, X.Y. & Bunke, H. (1989) Segmentation of the needle map of objects with curved surfaces. Pattern Recognition Letters 10: 181-187.

Kasif, S., Kitchen, L. & Rosenfeld, A. (1983) A Hough transform technique for subgraph isomorphism. Pattern Recognition Letters 2: 83-88.

Koivunen, V. & Pietikäinen, M. (1992) Experiments with combined edge and region-based range image segmentation. Theory & Applications of Image Analysis—Selected Papers from the 7th Scandinavian Conference on Image Analysis. World Scientific, Singapore.

Krishnamurthy, E.V. & Ziavras, S.G. (1988) Matrix g-inversion on the Connection Machine. Internal report, University of Maryland.

Lawson, C.L. & Hanson, R.J. (1984) Solving least squares problems. Prentice Hall.

Li, H. & Lavin, M.A. (1986) Fast Hough transform: A hierarchical approach. Computer Vision, Graphics, and Image Processing 36: 139-161.

Merlin, P.M. & Farber, D.J. (1975) A parallel mechanism for detecting curves in pictures. IEEE Transactions on Computers 28: 96-98.

Oshima, M. & Shirai, Y. (1983) Object recognition using three dimensional information. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5(4): 353-362.

Pieskä, S., Riekki, J. & Taipale, T. (1991) Controlling autonomous loading manipulator with Planning-Executing-Monitoring cycles. Poster, SPIE's Intelligent Robotic Systems Symposium, Boston.

Pong, T.C., Shapiro, L.G. & Haralick, R.M. (1981) A facet model region growing algorithm. Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing (PRIP): 122-139.

Pratt, V. (1987) Direct least-squares fitting of algebraic surfaces. Computer Graphics 21(4): 145-152.

Pulli, K. (1991) 3d graphics on the multiprocessor system DAMP. Master's thesis. University of Oulu, Department of Electrical Engineering, p. 72.

Pulli, K., Pietikäinen, M. (1993) Range image segmentation through fusing orientation and depth information. Submitted for Machine Vision and Applications.

Riekki, J. (1993) A control system for an autonomous machine. (in Finnish), Licentiate in Technology thesis. University of Oulu, Department of Electrical Engineering.

Riekki, J., Röning, J., Silvén, O., Pietikäinen, M. & Koivunen, V. (1991) Pick-and-place guidance utilizing an integrated control method and structured light ranging. Proceedings of the SPIE Intelligent Robots and Computer Vision X, SPIE vol. 1607: 689–699.

Rioux, M. & Cournoyer, L. (1989) The NRCC Three Dimensional Image Data Files. CNRC 29077, National Research Council Canada.

Röning, J., Riekki, J. & Kemppainen, S. (1990) Simulator for developing mobile robot systems. Proceedings of the Mobile Robots V, SPIE vol. 1388: 350-360.

Rousseeuw, P. & Leroy, A. (1987) Robust Regression & Outlier Detection. John Wiley & Sons, New York, p. 329.

Rousseeuw, P. & van Zomeren, B. (1990) Unmasking multivariate outliers and leverage points. With Comments and Rejoinder, Journal of the American Statistical Association 85(411): 633-651.

Sabata, B., Arman, F & Aggarwal, J.K. (1990) Segmentation of 3-d range images using pyramidal data structures. Proceedings of the 3rd International Conference in Computer Vision, pp. 662-666.

Shirai, Y. (1987) Three-Dimensional Computer Vision. Springer Verlag, Berlin, p. 297.

Shirai, Y. & Suwa, M. (1971) Recognition of polyhedra with a range finder. Proceedings of the 3rd International Joint Conference on Artificial Intelligence, pp. 80-87.

Shvaytser, H & Bergen, J.R. (1991) Monte Carlo Hough transforms. Proceedings of the 7th Scandinavian Conference on Image Analysis, pp. 80-87.

Sklansky, J. (1978) On the Hough technique for curve detection. IEEE Transactions on Computers 21:46-57.

Stockman, G.C. & Agrawala, A.K. (1977) Equivalence of Hough curve detection to curve matching. Communications of the ACM 20: 820-822.

Taylor, R.W., Savini, M. & Reeves, A.P. (1989) Fast segmentation of range imagery into planar regions. Computer Vision, Graphics, and Image Processing 45: 42-60.

Technical Arts (1989) User's Manual and Application Programming Guide. Preliminary Data, Technical Arts Corporation.

Tomita, F, & Kanade, T. (1984) A 3d vision system: Generating and matching shape descriptions in range images. Proceedings of the 1st Conference on Artificial Intelligence Applications, pp. 186-191.

Westman, T., Harwood, D., Laitinen, T. & Pietikäinen, M. (1990) Color segmentation by hierarchical connected component analysis with image enhancement by symmetric neighborhood filters. Proceedings of the 10th International Conference on Computer Vision and Pattern Recognition Systems and Applications, pp. 769-802.

Xu, L., Oja, E. & Kultanen, P. (1990) A new curve detection method: Random Hough transform (RHT). Pattern Recognition Letters 11(5): 331-338.

Yokoya, N. & Levine, M.D. (1989) Range image segmentation based on differential geometry: A hybrid approach. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-11(6): 643-649.

Zucker, S.W. (1976) Region growing: Childhood and adolescence. Computer Graphics and Image Processing 5: 382-399.