USACERL Technical Report P-90/27
September 1990

US Army Corps
of Engineers
Construction Engineering
Research Laboratory

Enhancements to a Guidance System for an Automated Warehouse Equipped With Mobile Robots

by
Shiv G. Kapoor
Satheesh R. Menon
Ram Pandit

In case of national mobilization, the U.S. Army must be able to construct large training camps within a very short time and with a severe shortage of skilled labor. Incorporating robots into the construction process appears to be an optimal solution to this problem.

The overall purpose of this research is to develop an autonomous warehouse system using mobile robots. The results of previous research directed toward this objective are documented in Technical Report P-87/07, *Development of a Guidance System for an Automated Warehouse Equipped With Mobile Robots*, by S.G. Kapoor, et al. (USACERL, February 1987).

This phase of research included development of preprocessors to input warehouse layout information. The sensor technology requirements of mobile robot navigation were analyzed. The control algorithms for navigation planning were modified to operate on a multirobot environment, and the control algorithms for navigation were enhanced to consider optimizing criteria for idling and loaded robots. Algorithms to interpret sensor data for bypassing, blocking, retracing, and robot interaction strategies under these situations were developed. The color graphic simulation model was updated to include the new algorithms. Recommendations for sensors and communication links are presented. A plan for future system development is proposed.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE August 1990	3. REPORT TYPE AND DATES COVERED Final

4. TITLE AND SUBTITLE
Enhancements to a Guidance System for an Automated Warehouse Equipped With Mobile Robots

5. FUNDING NUMBERS

6. AUTHOR(S)
Shiv G. Kapoor, Satheesh R. Menon, and Ram Pandit

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Construction Engineering Research Laboratory (USACERL)
PO Box 4005
Champaign, IL 61824-4005

8. PERFORMING ORGANIZATION REPORT NUMBER
USACERL P-90/27

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Construction Engineering Research Laboratory
PO Box 4005
Champaign, IL 61824-4005

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Copies are available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited.	

13. ABSTRACT (Maximum 200 words)
In case of national mobilization, the U.S. Army must be able to construct large training camps within a very short time and with a severe shortage of skilled labor. Incorporating robots into the construction process appears to be an optimal solution to this problem.

The overall purpose of this research is to develop an autonomous warehouse system using mobile robots. The results of previous research directed toward this objective are documented in Technical Report P-87/07, *Development of a Guidance System for an Automated Warehouse Equipped With Mobile Robots*, by S.G. Kapoor, et al. (USACERL, February 1987).

This phase of research included development of preprocessors to input warehouse layout information. The sensor technology requirements of mobile robot navigation were analyzed. The control algorithms for navigation planning were modified to operate on a multirobot environment, and the control algorithms for navigation were enhanced to consider optimizing criteria for idling and loaded robots. Algorithms to interpret sensor data for bypassing, blocking, retracing, and robot interaction strategies under these situations were developed. The color graphic simulation model was updated to include the new algorithms. Recommendations for sensors and communication links are presented. A plan for future system development is proposed.

14. SUBJECT TERMS	15. NUMBER OF PAGES
robots warehouses	77
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

FOREWORD

This project was funded by the Facility System (FS) Division of the U.S. Army Construction Engineering Research Laboratory (USACERL) under the Technical Director's initiative. The work was accomplished under contract by the Department of Mechanical and Industrial Engineering at the University of Illinois at Urbana/Champaign, IL. Members of the research team were Shiv G. Kapoor, Satheesh R. Menon, and Ram Pandit. Professor U.S. Palekar provided invaluable assistance and support. USACERL's principal investigator was Robert B. Blackmon. Michael J. O'Connor is Chief of USACERL-FS. The technical editor was Gloria J. Wienke, USACERL Information Management Office.

COL Everett R. Thomas is Commander and Director of USACERL and Dr. L.R. Shaffer is Technical Director.

CONTENTS

CONTENTS (Cont'd)

FIGURES

TABLES

ENHANCEMENTS TO A GUIDANCE SYSTEM FOR AN AUTOMATED WAREHOUSE EQUIPPED WITH MOBILE ROBOTS

1 INTRODUCTION

Background

Mobile robots and Automatically Guided Vehicles (AGVs) are now being used more often in material handling applications. In flexible manufacturing cells, AGVs offer the link between the factory floor and the material storage division. In warehouses, AGVs and mobile robots provide easy access to the storage and receiving areas. The use of mobile robots in building construction sites has been extensively studied.[1] The added degree of freedom and independent mobility of these devices make them more desirable for material transfer than conventional belt conveyors and stacker cranes.

Because AGVs are dependent on wires, tapes, or signals for guidance, mobile robots offer better flexibility and performance. Powered by microprocessors, the robot's decisionmaking modules select the control policies. Using mobile robots for material handling applications could prove to be a giant leap towards autonomous material transfer systems.

Previous research using autonomous mobile robots to transfer construction material involved planning and designing a construction warehouse and its components and developing a landmark system for robot guidance.[2] A modular layout was chosen for study. A standard bill of materials was used in designing the warehouse. The warehouse contained racks of palletized items that were grouped into modules separated by aisleways. A high-speed highway encircled the warehouse (Figure 1). The landmarks, which could be bar codes, patterns, or signals were placed on the walls at strategic (decisionmaking) locations along the pathways. The landmarks provided the information needed for the robot to confirm its location, correct its course, adjust its speed, determine its orientation, and other essential navigational functions.

The research also developed a path planning algorithm. The algorithm considered both static obstacles on the path and dynamic obstacles that may appear as the robot moves along its path.

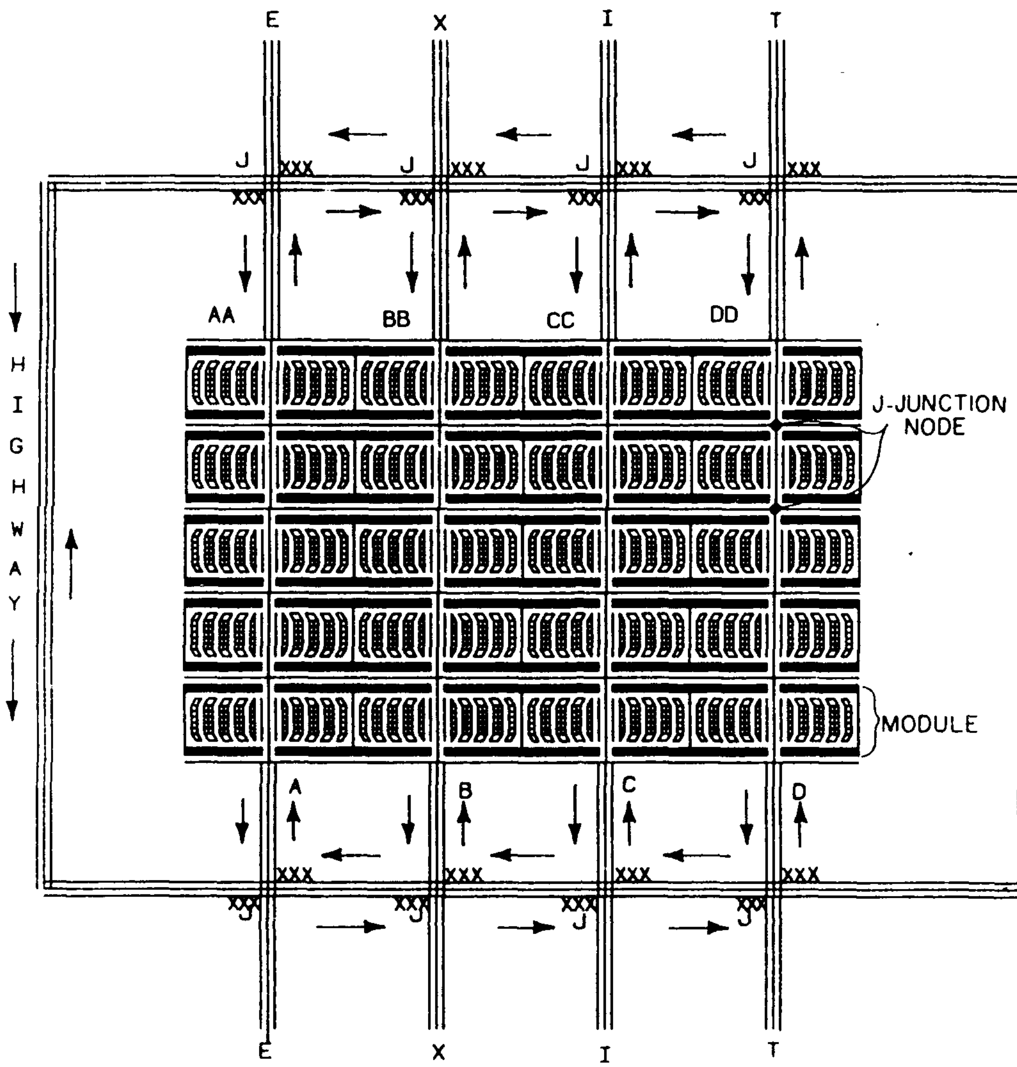
A network structure for the warehouse layout was developed. Nodes and junctions were defined as points where the direction of movement could be changed. The distance between nodes was assumed to be constant and the path from the start node to the target node was found using a breadth-first search technique.[3]

The collision avoidance technique in the system was based on the use of a node-time chart for each robot. Every node in a robot's path has an associated arrival time based on the robot's distance from the

[1] S.C-Y Lu, T.D. Brand, and S.G. Kapoor, *Sensors and Guidance Technology Related to Mobile Robots*, Technical Report P-87/02/ADA182989 (U.S. Army Construction Engineering Research Laboratory [USACERL], June 1987).

[2] S.G. Kapoor, et al., *Development of a Guidance System for an Automated Warehouse Equipped With Mobile Robots*, Technical Report P-87/07 (USACERL, February 1987).

[3] N. Nilsson, *Principles of Artificial Intelligence* (Tioga Publishing Company, 1983); P.H. Winston, *Artificial Intelligence* (Addison-Wesley, 1984).



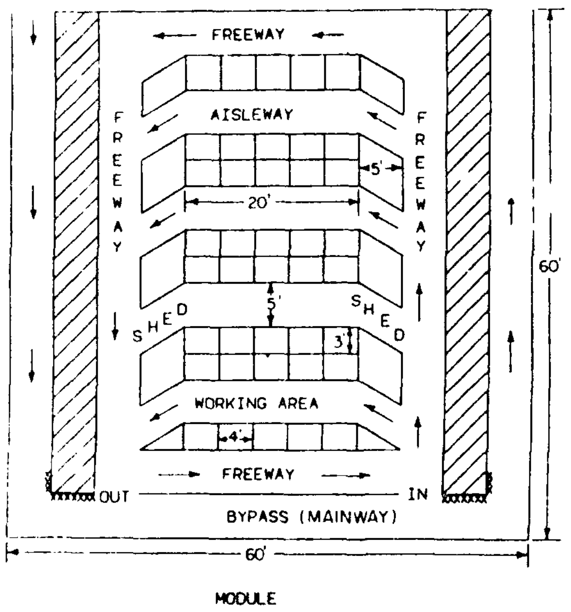


Figure 1. Warehouse layout and module.

node and average velocity. When arrival times of different robots at a particular node fall within a specific tolerance limit, a collision is possible at that node. Collision is avoided by altering the arrival time of the lower priority robot.

In the final stage of the previous research, a simulation model using animated color graphics was developed to demonstrate the algorithms.

Objectives

The overall objective of this research is to develop an autonomous warehouse system using mobile robots. The goals of this part of the project were to:

1. Develop a user-friendly preprocessor to define the warehouse layout based on specified sizes for the storage areas and aisles,

2. Investigate the technological requirements for remote sensing, landmark identification, positioning, and other guidance operations,

3. Enhance the algorithms for path planning (using an A* search) to operate on a multirobot environment where the nodes are not always equidistant and the velocities in the pathways vary,

4. Enhance the control algorithms for navigation to consider optimizing criteria for idling and loaded robots,

5. Develop algorithms to interpret sensor data during bypassing, blocking, retracing, and robot interactions,

6. Revise the simulation model to incorporate the enhanced algorithms and a status monitoring capability,

7. Select sensors for navigation,

8. Evaluate communication links, and

9. Develop a plan to continue the work through physical demonstration of the system for automating warehouse operations.

Approach

The first step in this phase of research was to design and implement the warehouse preprocessor. Chapter 2 describes the two types of layout creation. The node generation schemes, including the nodal representation of the world, is also explained in detail.

Details of the operational requirements and characteristics of the sensors needed for mobile navigation are presented in Chapter 3. The result of a market survey of available sensors is also presented.

The next step was to conduct a literature search on control algorithms for navigation planning and develop algorithms to control a multirobot environment using sensors for bypassing and collision avoidance while bypassing. This step is discussed in Chapter 4. Selecting the optimum paths based on the heuristic A* search is also discussed in Chapter 4. The different optimizing criteria are explained and examples are presented.

Revisions to the graphic simulation model are discussed in Chapter 5. This discussion includes information on the control structure architecture of the decisionmaking modules for navigation and selecting communication links between robots and the central computer.

Chapter 7 presents a summary of the research. Recommendations for a long range plan to continue the development of this automatic system are presented in Chapter 8.

Mode of Technology Transfer

Upon completion of the final algorithms and definition of the optimal landmarks, the control system will be tested and demonstrated in an Army warehouse using commercially available mobile robots modified to accept the sensors and the complex programming developed in this project. A decision on field implementation and industry involvement in producing the needed mobile robots will be based on the outcome of the field test.

2 DESIGN OF WAREHOUSE LAYOUT

The simulation model has been enhanced to include two preprocessors. Preprocessor I can be used to enter essential information to either describe the *existing warehouse or to design a layout of bins and* racks within existing space. In addition to other benefits discussed below, the preprocessor provides a systematic way to enter data (also discussed below). Preprocessor II can be used to determine the optimum floor area and storage rack/bin layout based on the materials to be stored in pallet sizes. In each preprocessor, the system considers the space needed to operate forklifts around the storage units. The model uses the same assumptions used in the control system. The autonomous forklifts (robots) will navigate using on-board sensors to read landmarks located near each pathway intersection. The landmarks could be wall-mounted patterns, bar codes, signals, etc., that provide the information needed to confirm the robot's location, or serve as the basis for course correction, speed adjustment, determining the robot's orientation with respect to the wall, or other essential navigational functions. Using this information, the system can create a global environment in which the robot can operate.

Preprocessor I

This preprocessor allows the user to study the navigational policies in an existing warehouse or to design a layout by specifying rack sizes and their orientations. The preprocessor is developed as a mini-CAD (computer aided design) system and allows the user to define five types of rack arrangements. Arbitrary placement of the racks, automatic node generation, defining pathways connecting node points, defining high-speed highways, and generating the node-node successor data base are some of the functions of the preprocessor. The working preprocessor I is explained in Appendix A.

Preprocessor II

In cases where the layout in which to develop the navigation policies does not exist, preprocessor II can be used to generate a layout according to user-specified warehouse capacity, storage rack dimensions, and pathway dimensions. This preprocessor designs the warehouse to be nearly square and provides the mathematical representation of the layout in terms of nodal coordinates and their connectivity. The procedure to design the warehouse layout is explained in Appendix B.

Nodal Representation of the Warehouse

The nodal generation scheme is based on the following steps:

1. Landmarks are placed at the entrance and exit points of each side of the block (Figure 2). Hence, there are four nodes associated with each block. These nodes are stored in array N and are numbered as:

$$N_{i,j} = 4*(i-1) + j,$$
$$i = 1 \ldots \mathrm{BNUM}$$
$$j = 1 \ldots 4$$

BNUM is the number of blocks in the warehouse. The node represented by $N_{i,j}$ is the jth node of ith block. The total number of nodes associated with BNUM blocks = 4*BNUM. In Figure 3, these nodes are shown as numbered 1 to 100. This numbering is internally generated by the computer and corresponds to the numbering in Figure 2. For example, node number 1 is reached when i=1 and j=1. It is also referred to as the first node (j=1) of block 1 (i=1) and is stored in array N at location (1,1). Hence, $N_{1,1}=1$.

2. A robot from the highway is allowed to enter the blocks only through the alleys because the aisleways may be occupied with robots picking or placing items. Landmarks placed at the entrance and exit points of blocks adjacent to the highway are used to connect the alleys with the two-lane highways. Two landmarks (one for each lane) are needed to connect one alley of a block with the highway. Since a block has two alleys, four landmarks are needed on the highway for each block. If a number of blocks (P1) are in one row, the number of nodes required on the highway to connect one side of the layout with the highway = 4*P1.

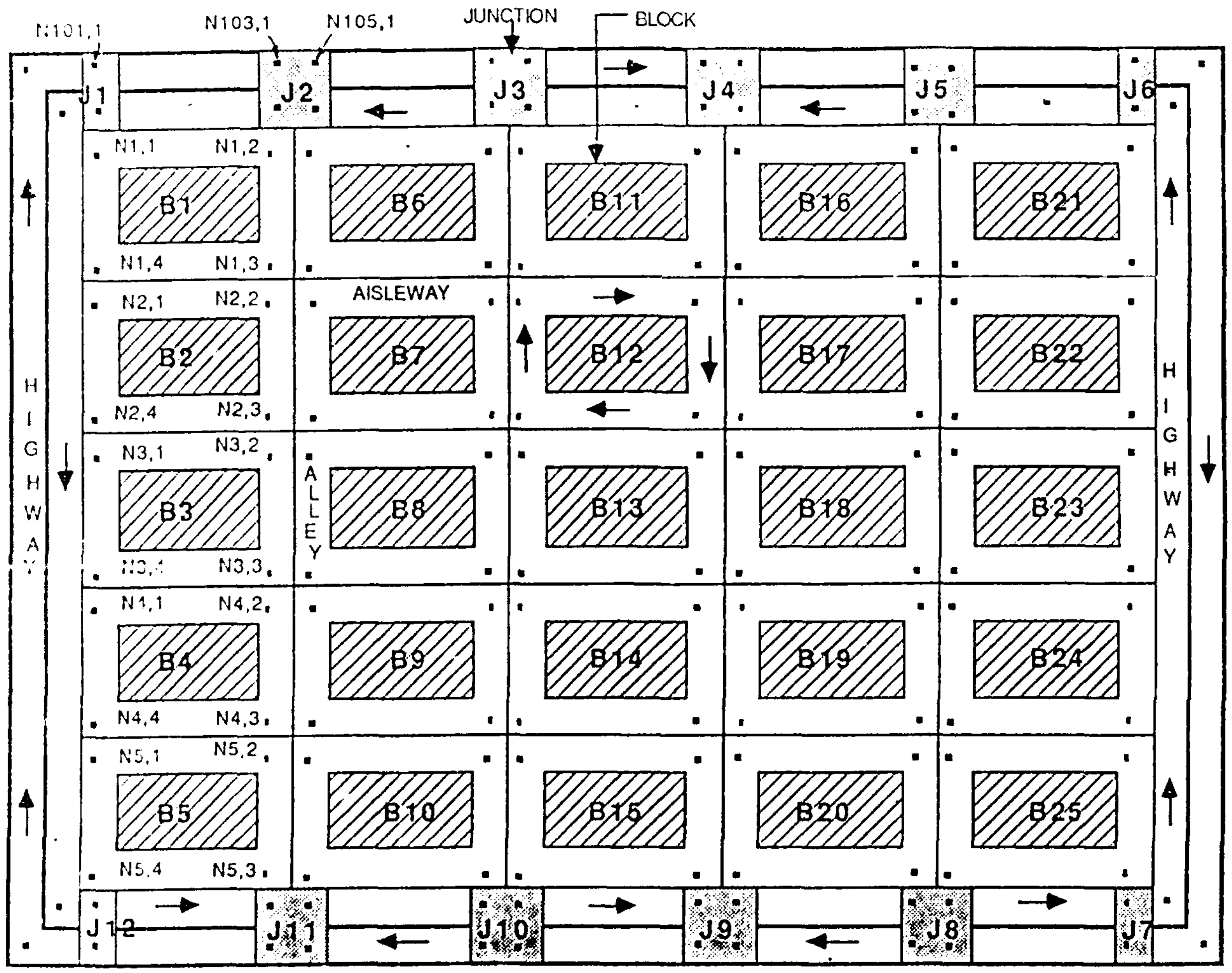


Figure 2. Nodal representation of the warehouse in preprocessor II.

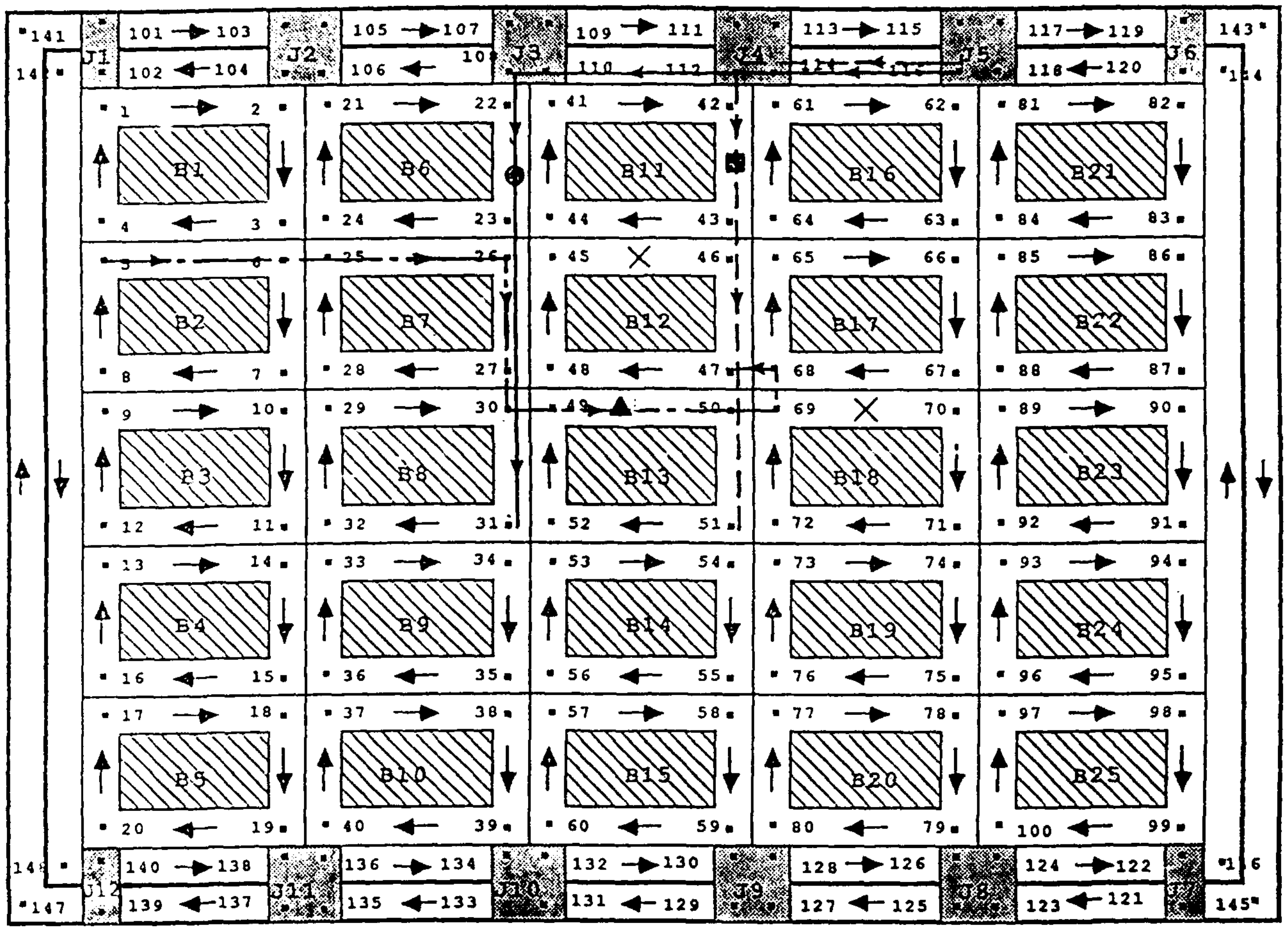


Figure 3. Node number generation.

Considering the top and bottom portions of the layout, the total number of nodes on the highway = 2*(4P1) = 8*P1. These nodes are numbered clockwise starting with the top left aisleway. These nodes are stored in array N and are numbered as:

$$N_{(4*BNUM)+j,1} = 4*BNUM + J$$

where J = 1 . . . 8*P1.

In Figure 3 these nodes are numbered 101 through 140.

3. Another set of nodes is needed on the highway to indicate the turns. Since the highway has four turn points, and two landmarks are needed at each corner, eight landmarks are needed. These nodes are stored and numbered as:

$$N_{(4*NUM\ +\ 8*P1)\ +\ J,1} = (4\ +\ BNUM\ +\ 8*P1)\ +\ J$$

where J = 1 . . . 8.

In Figure 3 these nodes are numbered 141 through 148.

The total number of landmarks needed for a layout which has a highway surrounding P1 blocks in each row and Q1 blocks in each column can be given as the sum of the total number of nodes associated with blocks, the total number of nodes associated with the highway and alley junctions, and the total number of nodes needed to indicate turns in the highway, or (4*P1*Q) + (8*P1) + 8.

Determining Coordinates

The dimensions of the warehouse are determined based on the user-specified dimensions of blocks, width of aisles and alleys, and width of the highway lanes.

Length of the warehouse:

$$\text{LENGTH} = \text{P1} + \text{BLENGTH} + 2*\text{P1}*\text{ASL1} + 2*\text{HLANE}$$

where BLENGTH = LENGTH of each module

ASL1 = alley width

HLANE = width of highway lane.

Width of warehouse:

$$\text{WIDTH} = \text{Q1}*\text{BWIDTH} + 2*\text{Q1}*\text{ASL2} + 2*\text{HLANE}$$

where BWIDTH = WIDTH of each module

ASL2 = WIDTH of aisle.

Once the dimensions of the warehouse are known, the node coordinates can be determined by considering any corner point of the warehouse as reference. For example, considering the lower left corner in Figure 2 as (0,0) in a Cartesian coordinate system, coordinates of node $N_{1,1}$ are determined as:

$$X = 2*\text{HLANE} + \text{ASL1}/2$$

$$Y = \text{LENGTH} - 2*\text{HLANE} - \text{ASL2}/2.$$

Similarly, the rest of the coordinates can also be determined.

Motion Rules Between Nodes

Once the placement of landmarks is determined, the next step is to determine the direction of movement between nodes and the connectivity of the nodes to form paths. A set of rules using the layout model information to decide the direction and connectivity between nodes has been developed. These rules are as follows:

1. Connectivity within a block. The nodes associated with any block i are connected in a clockwise direction (Figure 2). The relationship can be expressed as:

$N_{i,1} \rightarrow N_{i,2}$

$N_{i,2} \rightarrow N_{i,3}$ for i = 1 . . . BNUM

$N_{1,3} \rightarrow N_{1,4}$

$N_{i,4} \rightarrow N_{i,1}$

2. Connectivity between adjacent blocks. If block i and block k are adjacent blocks in a row, then the connectivity is given by:

$N_{i,2} \rightarrow N_{k,1}$

$N_{k,4} \rightarrow N_{i,3}$

If block i and k are adjacent blocks in a column, then the connectivity is given as:

$N_{1,3} \rightarrow N_{k,2}$

$N_{k,1} \rightarrow N_{i,4}$

3. Connectivity between highway junctions and block alleys. For robots to access modules and the highway, connectivity between alleys of the blocks adjacent to highways and highway junction nodes must be defined. The adjacent blocks are defined as top side blocks and bottom side blocks. In Figure 2, blocks B1, B6, B11, B16, and B21 are top side blocks and B5, B10, B15, B20, and B25 are bottom side blocks. Accordingly, $N_{1,1}$, $N_{1,2}$, $N_{6,1}$, $N_{6,2}$, $N_{11,1}$, $N_{11,2}$, $N_{16,1}$, $N_{16,2}$, $N_{21,1}$, and $N_{21,2}$ form top alley nodes and $N_{5,4}$, $N_{5,3}$, $N_{10,4}$, $N_{10,3}$, $N_{15,4}$, $N_{15,3}$, $N_{20,4}$, $N_{20,3}$, $N_{25,4}$, and $N_{25,3}$ form bottom alley nodes. The connectivity is defined as follows:

Connectivity between junction nodes and top alley nodes:

$N_{\ell,1} \rightarrow N_{j+2,1}$

$N_{j+4,1} \rightarrow N_{\ell,2}$

where i = 1 . . . P1

$\ell = Q1(i-1) + 1$

$j = 4(i-1) + 4*BNUM.$

Connectivity between junction nodes and bottom alley nodes:

$N_{\ell,3} \rightarrow N_{j+2,1}$

$N_{j+4,1} \rightarrow N_{\ell,4}$

where $i = P1 \ldots 1$

$\ell = i*Q1$ (represents the block number)

$j = 4*(P1-1) + 4*P1*Q1+4*P1$ (gives the offset to the junction node).

4. Connectivity between highway nodes. For movement to be possible on the highway, the connectivity between highway nodes must be defined. The node corresponding to junctions J1 . . . J6 in Figure 2 are top side highway nodes, and nodes corresponding to junctions J7 . . . J12 are bottom side highway nodes. The connectivity is defined as follows.

(a) Top side:

$N_{j,1} \rightarrow N_{J+2,1}$

$N_{k} \rightarrow N_{k-2,1}$

where $i = 1 \ldots (2*P1-1)$

$j = 2*i-1 + 4*P1*Q1$

$k = 2*(2*P1 - i+1) + 4*P1*Q1.$

(b) Bottom side:

$N_{j,1} \rightarrow N_{j+2,1}$

$N_{j+3} \rightarrow N_{j+1,1}$

where $i = 1 \ldots (2*P1-1)$

$j = (2*i-1) + 4*P1 + 4*P1*Q1.$

5. Connectivity between highway junction nodes and turning point nodes. Since the warehouse is rectangular and the highway surrounds it, the highway has four turn points. These turn points need eight landmarks for robot navigation. These eight nodes are to be connected with the adjacent junction nodes and turn point nodes. This requires 12 connections. Those connections can easily be defined according to the movement directions as shown in Figure 3.

3 SENSOR REQUIREMENTS AND CURRENT TECHNOLOGY

Sensors play a major role in the robots' navigation system. The local planning, or microplanning, is entirely dependent on the output of the sensor module as shown in Figure 4. The sensor control module decides which type(s) of sensor(s) to activate. The information is processed, integrated, and sent back to the navigation module. Sensor data interpretation is carried out in the navigation module and the movement strategy is passed on to the robot's pilot.

Sensor Requirements

The robot can interact with the environment through the use of sensors that can be broadly classified by the following applications:

1. Sensors for guidance in mainways and highways,

2. Sensors for guidance in the modules, and

3. Sensors that feedback information regarding robot functions.

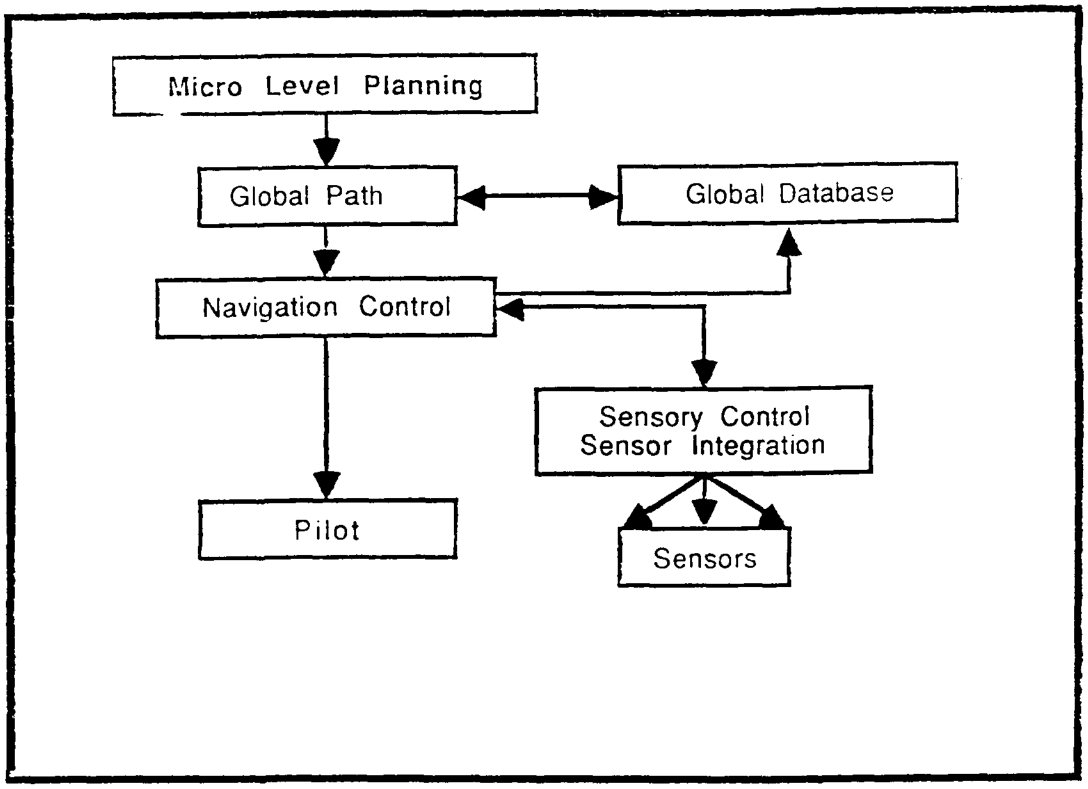


Figure 4. Sensory control for local planning.

Sensors for mobile robot navigation should provide maximum information from a minimum number of landmark readings. The prevailing sensor technology requires static readings. In other words, the robot has to stop each time the sensor takes its readings. When multiple sensors are involved, a hierarchical ordering of the sensors will have to be made based on microprocessor considerations. The restrictions imposed on sensors make them an important research issue in mobile navigation.

Sensors for Guidance in Mainways and Highways

When the central computer receives an order, it plans a path for the robot. The robot needs a position referencing sensor to check for the landmark corresponding to the starting node designated by the computer. A landmark tracking sensor is needed to adjust the robot's orientation with respect to the landmark. If the landmark is a pattern, the robot needs a vision system to analyze it. Bar code landmarks could be read using infrared readers or lasers. Pattern or code analysis is a compare and check function. Since the robot knows the planned path, it compares the landmarks it finds to those expected from the path.

After determining its position, the robot scans in the direction of the next node of its path. The scanning sensor should be able to track the position of obstacles. A range finding sensor would be highly desirable for this application if it could operate in a dynamic environment and measure the velocity of moving objects while the robot is also moving.

When a loaded robot must turn a corner, it would be economical to rotate the wheels without rotating the chassis.[4] The amount of angular motion needed could be determined using a gyroscopic sensor.

Movement along a straight path almost always results in the robot straying from the course. A system requiring the robot to follow fixed shapes for course correction has been developed.[5] Because the shapes must be displayed in the robot's path, this system may not be practical for rough terrain. The landmark tracking sensor could be used to a certain extent to correct deviations, but if the deviations are large, it may be better to use an orientation maintaining sensor to ensure travel along the right course.

The mobile navigation system for robots in an automated warehouse is based on a prelearned environment or world (by defining fixed pathways and restricting movements in certain directions). However, cases may arise when the robot is required to construct a local map within the bounds of the world. For example, the robot will need a map if it is entangled in a maze of obstacles and has to plan a way around them. This requires a scene analysis to be carried out. An accurate but time-consuming analysis could be done using visual sensors. Sonic sensors can analyze faster, but the result is less dependable.

Sensors for Guidance in the Modules

Pick and place operations are carried out in the aisleways of the modules. Sensors are needed to identify the pallets. Bar codes and infrared readers may be used for this purpose. The robot's position in the modules could be determined using proximity sensors. Also, tactile sensors would help to limit pressure in grasping operations.

[4] J. Imai, Y. Kawashima, and K. Hironaka, "Advanced Automated Transportation System," *Procedures of the First International Conference on Automation in Warehousing* (Mitsubishi Electric Corporation, Japan, 1975), pp 175-184.

[5] K. Komonya and K. Tamie, "Research Review for Realization of Autonomous Locomotion," *Materials Flow*, Vol 3 (1986), pp 3-15.

Sensors for Robot Functions

The types of functional sensors needed depend on the type of robot and may include velocity transducers, force transducers, fuel/power readers, gyroscope, and steering control. A discussion of the sensors for monitoring the robot's operating systems is beyond the scope of this research.

The sensors could also be classified based on their intended functions. For navigation in a known environment, the sensors may need to:

1. Confirm the position of the robot in the global map,

2. Determine the reference position for local planning, and

3. Detect obstacles and objects.

Sensors for Position Confirmation

Global navigation is based on the network model of the world. Movement is from one node to the next node of the robot's path. To ensure that the robot does not stray to a node other than its intended node, landmarks must be placed at the node points. The landmark may have characteristics specific to the node it represents, including codes to denote the name/number of the node, the names/numbers of the successor nodes and their orientations, etc. The landmarks may be patterns, signals, or bar codes. Patterns require a vision system for image processing. The information that can be contained in patterns is limited. Signals or beacons offer a flexible marker system. Like patterns, signals can only supply positional information and there may be problems in transmitting information due to background noise. Bar codes seem to offer a robust landmark system. They are available in different sizes and the amount of information is not limited. Bar codes can be easily read using a scanner and decoder.

Sensors for Position Referencing

While moving between nodes, the robot is solely dependent on the sensors for navigation. Different sensors may supply information for local decisionmaking and this data needs to be integrated by the sensor module. Collected sensor data needs to have a reference point. For example, when the robot's sonic sensors discover an obstacle, the sensors provide only the range of the obstacle from the robot's reference plane. This information will then have to be converted into the world reference coordinates. For this, the robot needs to know the distance it has traveled with respect to the last landmark node. Position referencing could be achieved by a simple dead-reckoning system such as encoders.

Sensors to Detect Obstacles

Obstacle detection is the main activity performed by the robot while moving. Before moving from a node, the robot has to ensure that there are no obstacles in its path to the next node. The obstacle may be a static one (dropped items, etc.) or a slow moving robot. Since scanning is done from node to node, sensors that can cover large ranges are required. Sensors need to be activated at all times while the robot is moving to track moving obstacles. Proximity/short range sensors will be required for these tasks.

Vision, laser, and sonic sensors are some of the options available for obstacle detection. A vision system can become computationally expensive and is very much dependent on light intensity variations.

Lasers are recommended for scanning long distances and for edge detection. Ultrasonic sensors are simpler than other sensors and may satisfy the requirements.

Current Sensor Technology

Vision Systems

Vision systems are used for pattern identification, scene analysis, and ranging. Carnegie-Mellon University's (CMU's) Neptune, and Stanford University's Stanford Cart are mobile robots that use a vision system for navigation. Machine vision systems carry out the operations of image formation, preprocessing, analysis, and interpretation.[6] Illumination, sensing, and imaging are the three processes involved in image formation. Illumination could be done by back lighting, diffused or polarized front lighting, front lighting with directed dark or bright fields, or structured lighting. An electronic imager acts as a sensor, collecting light from a scene and converting that light into electrical energy using photosensitive targets. Vidicon cameras and solid state cameras (charge-coupled devices [CCDs] and charge-injected devices [CIDs] are commonly used for imaging. Although Vidicon cameras are able to very quickly provide a great deal of information about a scene, they tend to distort the image and are subject to "image burn-in" on the photoconductive surfaces. They are also easily damaged by vibration and shock. Solid-state cameras, on the other hand, are smaller and more rugged. They last longer and show less image distortion. However, they are more expensive.

Image preprocessing consists of digital conversion, windowing, and image restoration. Processing uses an analog/digital converter to change the analog voltage values for the image into corresponding digital values, producing an array of numbers that represents the light intensity distribution over the image area. Vision systems can be classified as binary or gray scale based on how the digitizing is done. Binary systems provide only 2 values, but gray scale systems permit up to 256 values for different shades of gray. The sophisticated gray scale system conversion requires an extremely powerful microprocessor. The amount of data to be processed could be reduced by windowing and image restoration techniques. Edge detection and run length encoding are other preprocessing operations.

Preprocessing is followed by image analysis. This is the decisionmaking stage that begins with an analysis of the simplest features and continues with the addition of complicated features until the image is clearly identified. The object's position, orientation, geometric configuration, and the distribution of light intensity over its visible surface are factors in this operation. Stadimentry (direct) imaging, triangulation, and stereo or binocular vision are used for positioning or determining the distance of the object from the camera. Orientation information is important when a robot has to position itself relative to a part and is accomplished by methods such as equivalent ellipse, connecting three points, light intensity distribution, and structured light. Image segmentation, image shape, and image organization help to arrive at the geometric configuration of the object.

The final operation of image interpretation compares the results of the image analysis with a prestored set of standard criteria. Machine vision deals in probabilities and its goal is to achieve a probability of correct interpretation as close to 100 percent as possible. Feature weighing and template matching are commonly used for image interpretation.

[6]"The Fundamentals of a Machine Vision System," *Vision Technology* (June 1986), pp 2-9.

Due to the vast amount of information that has to be processed with vision systems, mobile robots using vision sensors tend to be very slow unless they have access to a large computer. The present vision systems are very costly.

Sonar Range Finders

Sonar sensors have proved particularly useful in navigation because they are inexpensive and able to grossly cover large areas more rapidly than detailed visual processing systems.[7] Sonic sensors are commonly used as proximity sensors to avoid obstacles and to detect objects such as flat walls. Most low cost sonar devices functicn by sending a multifrequency or "chirp" sound pulse from a transducer outward in a cone-shaped wavefront. The difference between the time of emission and return is then used to estimate the distance based on how far the wave could travel in one-half the period.

The speed of sound in air is determined by the following:

$$S = \sqrt{gHRT}$$

where H = ratio of specific heats of air at constant pressure and constant volume,
g = acceleration due to gravity,
R = gas constant for air,
T = temperature (in degrees Rankine).

The sonar calculated distance is determined by the following:

$$D_s = \frac{1}{2} ts$$

where t = time between emission and reception of sonar pulse and
s = speed of sound in air.

The actual distance is determined by the following:

$$D_A = D_S \sqrt{A_t / S_t}$$

where A_t = actual outside temperature in degrees Rankine and
S_t = standard temperature in degrees Rankine.

Ultrasonic transducers have recently gained popularity. However, various sources of error are associated with sonar range finders. Errors may be introduced because the speed of sound depends on the air

[7]C. Jorgensen, W. Hamel, and C. Weisbin, "Autonomous Robot Navigation," *BYTE* (January 1986), pp 223-235.

temperature and the surface characteristics of the object being scanned. The beam may be either too narrow or too wide to accurately scan an object. Errors are also introduced if mirror-like materials are being scanned.

The angular resolution of ultrasonic ranging systems can be increased by using multielement ranging arrays. Ultrasonic sensors using a cylindrical beam instead of a conical beam have recently been introduced into the market. These sensors have overcome many disadvantages associated with sonic sensing and are more accurate.

Wheel Encoders

Wheel encoders have been used as position estimators in robotic applications. The encoders are usually mounted on the motor shafts and the robot's position is estimated by converting the trajectory integration of encoder steps to world coordinates. Optical shafts, resistive sliding contacts, magnetic saturation devices, and proximity probes are some of the common wheel encoders. They are susceptible to errors due to wheel slippage.

Contact Switches

Contact switches are generally used as proximity sensors for collision avoidance. They provide binary information by making or breaking a circuit depending on whether or not the robot is in contact with an object.

Magnetic Proximity Sensors

Magnetic proximity sensors generate a magnetic field around the robot using coils driven by oscillators. They detect metallic objects in the vicinity of the field by inducing eddy currents in the object and measuring the frequency of oscillating (which depends on the distance to the object). Depending on the strength of the magnetic field, they could also be used for short distance ranging.

Infrared Sensors

Infrared sensors are used as proximity sensors, goal-seeking sensors, and motion detectors. They work on the principle that every object emits energy in the form of infrared radiation. These sensors could be altered to detect the wavelength emitted by objects (including humans) that may appear in the robot's path. Their performance tends to be erratic, especially in a crowded environment.

Laser Range Finders

Laser range finders are accurate sensors for determining long distances. Ranging is done by measuring the time of flight of a pulse of light. They require expensive retroreflection targets and are usually used for distances greater than 20 in. (0.5 meters). LIDAR (Light Detection And Ranging) is a popular laser range finder.

Level Sensors

Gyroscopic level sensors measure pitch and roll. Because they provide information about the steepness of slopes, they could be used outside and over rough terrain.

Goal-Seeking Sensors

Goal-seeking sensors guide the robot toward targets emitting infrared or sound waves of certain frequency and are generally used in obstacle avoidance.

Tactile Sensors

Tactile sensors are basically strain gauge sensors that detect the presence or absence of an object by contact. Magnetoelastic ribbons, optical fibers, potentiometers, and linear variable displacement transformers (LVDTs) are commonly used as tactile sensors. Mobile robots with tactile sensors have been used to determine the shape of obstacles.

Sensors Currently Being Used

Most mobile robots use many types of sensors to relay navigational information. The Intelligent Mobile Platform (IMP) has a rotating depth sensor for ranging mounted on its top and incremental wheel encoders for estimating position. The Stanford Cart and CMU's Rover use a TV camera mounted on a pan/tilt/slide mount, Polaroid ultrasonic transducers, short range modulated infrared proximity sensors, wheel encoders, and contact switches. CMU's Neptune has a ring of 24 polaroid ultrasonic transducers and a two-camera stereo vision system. The Terregator (also by CMU) uses shaft encoders to count wheel turns and a Grinnel digitizer with a black and white camera for vision analysis. Hilare (developed in France) uses a laser range finder, optical shaft encoders, and an ultrasonic system for obstacle avoidance. "Hermes-I" (developed by the Center for Engineering Systems Advanced Research) contains two ultrasonic ranging sensors and a solid state camera.

The PEGASUS system (developed by the University of Tennessee) uses contact switches for obstacle avoidance, 12 sonar transducers for object detection, and a gimbal-mounted fish eye camera and wheel encoders for position estimation.

Sensor Selection

Landmark Detection and Identification

Landmarks are located at every node point of the designed layout. Since the nodes indicate decision points, the landmarks should contain the following information:

- Current node.
- Total number of successors.
- First successor.
- Direction of first successor (straight, left, right, etc.).
- Second successor.
- Direction of second successor, etc.

Successors are those nodes to which the robot can move from its current node. Bar codes should be used as landmarks. These codes should be placed at a consistent height and at the nodal location as shown in Figure 5. Each bar code may have an associated beacon (infrared) to guide the robot to the landmark. The beacons on one side of the mainway may have a different emission frequency from those on the opposite lane. This avoids possible interaction or noise from the other lane and prevents the robots moving in one lane from straying into the other lane. Robots should be equipped with beacon detectors for seeking the landmark position and bar code readers consisting of a moving beam scanner and a decoder.

Position Referencing

Wheel encoders should be used to measure the distance traveled and to increase the reliability of the dead-reckoning system. Wheel encoders are subject to errors due to wheel slippage and cannot be the only method for measuring distance. Also, landmarks may pose problems due to incorrect orientation, damage, etc. By interacting with both the wheel encoders and the landmarks, the robot can ascertain its position and correct for any accumulated errors.

Wheel encoders are mounted on the rotating shaft. An absolute encoder may be sufficient. When a robot starts, the node number and information about successor nodes are read from the landmark. The wheel encoder is reset to zero distance at the landmark location. After the robot moves a specific distance from that location, the beacon detectors are activated and the next landmark is searched and read.

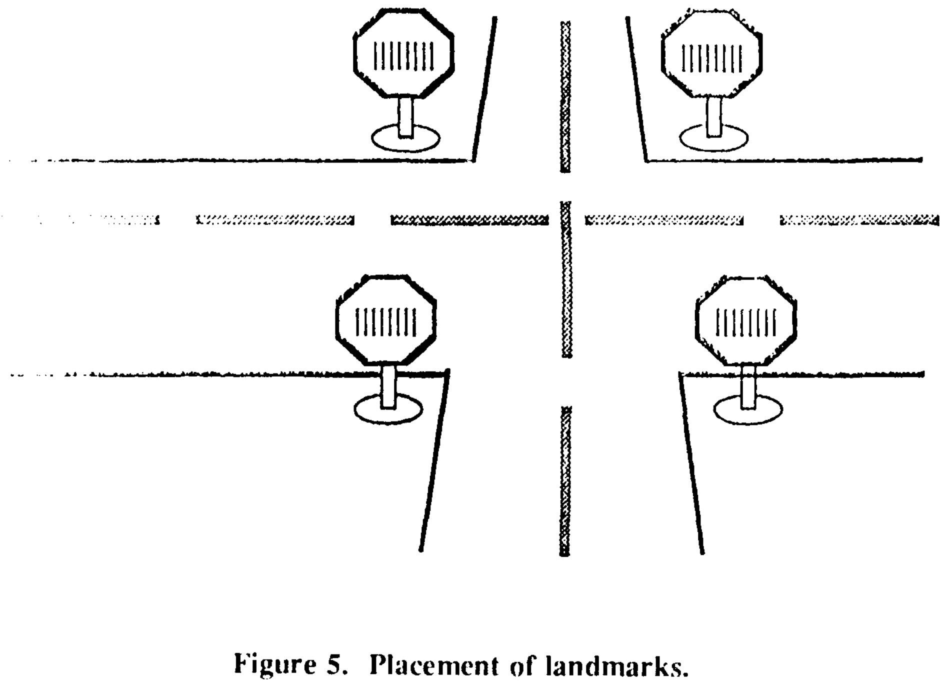


Figure 5. Placement of landmarks.

Obstacle Detection

Ultrasonic arrays and sensors are recommended for long range and short range obstacle detection. With arrays, it is possible to expand the area of coverage by increasing the number of arrays instead of the cone angle. The short range sensors need to track only the outline or envelope of the obstacle since the decision to bypass a block is made by scanning for free spaces in the pathways.

A market survey has been conducted on the various available sensors. The results of the survey are shown in Table 1.

Appropriate sensor systems are commercially available and should be integrated with the developed control system. Based on an understanding of the performance characteristics determined by physical testing, the control system can be modified to accept sensor input. The control system should be upgraded to handle and react to sensor input.

Table 1

Recommended Sensors for Navigation

Sensors	Company	Cat. No.	Price
Bar codes	Computype, Inc. Great Lakes Sales Office 433 W. Uni. Dr., Suite G MI-48063 (313)652-7123	(3" x 3", code 39)	$100/1000#s
Bar-code readers	Skan-a-matic P.O. Box S, Route 5 West Elbridge, NY-13060 (315)689-3961	S44020 Scanner D4100 Decoder B07184 Accs.	$2750.00 $1715.00 $82.50
Wheel encoders	Autotech Corporation 343 St. Paul Blvd. Carol Stream, IL-60188 (312)668-3900	Single turn Digisolvers	$625.00 (approx.)
Ultrasonic sensors	Polaroid Ultrasonic Components 119 Windsor Street Cambridge, MA-01219	#604142 Trans. #607089 Boards	$15.00/Unit $25.00/24#s

4 CONTROL ALGORITHMS

Algorithm Requirements

Previous research developed path planning algorithms to control individual robots. The complexity of the problem increases when planning paths for the number of robots expected in an automated warehouse. The multirobot navigational algorithm for a warehouse should meet the following requirements:

- Task assignments need to be prioritized based on the random demand for construction items.

- Path selection should be such that a loaded robot prefers the shortest distance path while an idling robot chooses the minimum time path.

- The robots should be able to communicate with each other as well as with the central computer.

- Sensing obstacles in the path should be continuous during navigation. When obstacles are tracked, the sensor data should supply information regarding the position of the obstacle, its orientation, and configuration.

- The sensor data should be analyzed and compared with the world map to determine the feasibility of going around an obstacle or retracing to an alternate path.

- The movement algorithm should include robot interaction rules. When two robots have to move along the same path, the algorithm should decide how to control the traffic to avoid collision. A simple decision rule may be for both robots to come to a halt and allow the higher priority robot to proceed while the lower priority robot waits. This may result in wasted time due to sensing and establishing communication. A smooth operation requires a collision avoidance algorithm that can look ahead for possible interaction and alter robots' velocities accordingly.

Most of the path planning algorithms use a network structure to model the global map. When obstacles are known beforehand, they are represented as specific geometric shapes and paths are planned around them. The paths are arbitrarily selected and the details of the obstacle's features are stored in the computer's memory.

The working environment of the robots is classified as "unstructured" or "structured." If knowledge about the world is incomplete due to the presence of unknown objects and exposed places, the world is said to be unstructured. A structured environment will have a priori information about the world. Path planning has been attempted in unstructured environments by building abstract sensor models of the world and searching for a path to the goal. This process is very time consuming.

For most practical material handling applications involving automation, in either a highly organized warehouse or a totally disorganized construction site, it is desirable to evolve a structured world from the original plan or layout. Exploration of unknown places needs to be limited since creating a map from sensor data is so time consuming that it is prohibitive in real world applications where time is an important factor in achieving efficiency. Robot navigation for material handling is simplified by prespecifying fixed routes for traffic movement.[8] In other words, traffic lanes could be defined and robots could be confined to move along these lanes. This eliminates the major problem of calculating "free space."

The obstacle avoidance routines generally try to avoid obstacles by tracking and bypassing them. These algorithms slow the robots's movement and require sensor information from various points. When pathways are fixed and the location of obstacles is known, searching for alternate paths that avoid the obstacles would be more efficient than bypassing them. Also, in situations when unexpected obstacles are encountered, the decision to bypass the obstacle or find an alternate path could be made easily by scanning only the outline of the obstacle in the pathway and calculating the feasibility of moving around it. Decisionmaking in dynamic environments needs to be studied in detail.

Literature Survey

An extensive literature survey on mobile navigational algorithms was conducted for related research.[9] The survey, however, dealt mainly with path planning algorithm development. Since obstacle tracking, obstacle avoidance, and robot interactions play a major role in navigation, the literature survey was expanded to emphasize these factors.

The find-path problem in navigational planning involves finding a collision-free path (preferably the optimal path) through a terrain that is arbitrarily populated with obstacles. The algorithms assume that a complete global model of the obstacle-laden environment is known. Many researchers model the obstacles and the free space as precise mathematical and geometrical entities. The methods include the generalized cone approach of Brooks, the sweeping volume method of Lozano-Perez, and the configuration space approach of Gouzenes.[10] The configuration space method, for example, divides a navigation space into zones that the reference point of a robot can occupy without the robot colliding with any obstacles. Paths that make maximum use of open areas are then defined from the reference point to the specified goal. In research with the CMU Rover, Moravec bounded all obstacles with circles that were enlarged to assure clearances for the robot edges.[11] Paths were then calculated as tangents to these circles. Other approaches involve either local or global navigation strategies. Local navigation deals with the immediate problems of obstacle avoidance whereas global navigation considers larger regional information like the plan of a building or long-term goals. Crowley describes global path planning in terms of a previously stored network of places.[12] Global navigation is defined as traversals along "legal highways" in known areas, with local movement based primarily on avoidance procedures using sensors. Crowley's local obstacle avoidance method plans a sequence of straight line paths that will take the robot around an unexpected obstacle, one to the left of the obstacle and one to the right. Tests are then made to see if either path to the goal is free in the composite local model. If either path is blocked, the procedure is called recursively to see if it is possible to get around the blockage.

[8] R.R. Lasecki, "AGVS: The Latest in Material Handling Technology," *CIM Technology* (Winter 1986), pp 90-94.
[9] S G. Kapoor, et al.
[10] R.A. Brooks, "Solving the Find-Path Problem by Good Representation of Free Space," *IEEE Transactions on System, Man and Cybernetics*, Vol SMC-13, No. 3 (Institute of Electrical and Electronics Engineers, 1983), pp 190-197; T. Lozano-Perez and M.A. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Communications of the ACM*, Vol 22, No. 19 (Association for Computing Machines, 1979), pp 560-570; L. Gouzenes, "Strategies for Solving Collision-Free Trajectories Problem for Mobile and Manipulator Robots," *International Journal of Robotics Research*, Vol 3, No. 4 (1984), pp 51-64.
[11] H.P. Moravec, "The CMR Rover," *Proceedings of the National Conference on Artificial Intelligence* (American Association of Artificial Intelligence, August 1980), pp 377-380.
[12] J.L. Crowley, Navigation for an Intelligent Mobile Robot," *IEEE Journal of Robotics and Automation* (1985), pp 31-41.

Keirsey and others describe a similar obstacle avoidance procedure using a global path plan, a local path navigator, and a movement-executing pilot.[13] The algorithm covers both known and unknown terrain. A vision system is used to trace obstacles. The algorithm operates either in a distinct or continuous mode and uses a heuristic search strategy. An interesting feature of this algorithm is that it updates the map based on sensor data.

Oliver and Ozguner describe a navigation algorithm for a vehicle equipped with a laser range finder sensor.[14] Region classification, path planning, path smoothing, and path execution are the four parts of the algorithm. Based on sensor data, regions are classified as peaks, pits, flat areas, unsafe hills, etc. A path is then planned by generating subgoals that define a series of line segments from the robot to the goal. Generating these subgoals involves a search that is directed by a cost function.

A road map production system model using a bidirectional search algorithm that yields a shortest path solution has been proposed by Siy.[15] The algorithm enables the robot to search quickly for alternate paths in the event it encounters an unexpected obstacle.

Krogh and Thorpe propose an integrated path planning and dynamic steering control for autonomous vehicles in uncertain environments.[16] Path relaxation is used to compute critical points along a globally desirable path using a priori information and sensor data.[17] Path relaxation is a two-step process that tries to find the path with the lowest total cost. The cost of a path is a combination of several factors, including distance traveled, nearness to objects, traversability of the terrain, and uncertainty about the area. The first step of path relaxation finds a preliminary path on an eight-connected grid of points. (The grid mesh size can be as large as the minimum dimension of the robot and still guarantee that no path will be missed.) The next step is to assign costs to paths on the grid and use an A* search to determine the best path along the grid from start to goal.[18] The second step of path relaxation fine-tunes the location of each node on the path to minimize the total cost. Each node's position is adjusted in turn (using only local information) to minimize the cost of the path sections on either side of the node. Since moving one node may affect the cost of the paths to its neighbors, the entire procedure is repeated until no node moves farther than some predetermined small amount. In the dynamic steering control algorithm for an uncertain environment, the steering control is specified as a feedback law and is based on the generalized potential field approach. The potential fields, which are both position and velocity dependent, eliminate the possibility of "stalling" at local minima in the potential fields. Functions are computed for goal attraction potentials and obstacle potentials. Khatib proposes using the artificial potential field concept for obstacle avoidance.[19]

[13] D.M. Keirsey, et al., "Algorithm of Navigation for a Mobile Robot," *IEEE International Conference on Robotics and Automation* (1984), pp 574-583.

[14] J.L. Oliver and F. Ozguner, "A Navigation Algorithm for an Intelligent Vehicle With a Laser Range Finder," *IEEE International Conference on Robotics and Automation* (1986), pp 1145-1150.

[15] P. Siy, "Road Map Production System for Intelligent Mobile Robot," *IEEE International Conference on Robotics and Automation* (1984), pp 562-570.

[16] B.H. Krogh and C.E. Thorpe, "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles," *IEEE International Conference on Robotics and Automation* (1986), pp 1664-1669.

[17] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *IEEE International Conference on Robotics and Automation* (1985), pp 500-505.

[18] N. Nilsson, *Principles of Artificial Intelligence* (Tioga Publishing Company, 1983).

[19] O. Khatib.

Enhancement of the Path Planning Algorithm

A performance study on the algorithm developed during previous research showed a need to optimize the path selection process. Use of the breadth-first search technique was not realistic and loaded robots may have different speeds than unloaded robots. The simulation study revealed a need to monitor the status of the system. With these factors in mind, both the planning algorithm and the simulation software were enhanced.

The path search process was optimized using the criteria of a minimum time path for idling robots and a shortest distance path for loaded robots. Since time and distance information are needed for optimization, the nodal data base needed to have the world coordinates of the nodes and the allowable velocity in the various links formed by the nodes. The search technique was changed to an A* search.[20]

A* search is an improved branch-and-bound search with an estimate of the remaining distance combined with the principle of dynamic programming. If the estimate of the remaining distance is a lower bound on the actual distance cost, then according to the admissibility condition, A* search produces optimal solutions.

The path planning algorithm needs to select the shortest distance paths for loaded robots and the minimum time paths for unloaded robots. The search therefore uses a cost function which is defined as:

Cost function F at a node

= cost to travel from the start node to the node (G)

+ cost to travel from the node to the goal node (H), or

$$F = G + H$$

The cost of H is a heuristic measure since selection of subsequent nodes to the goal are not known beforehand. Also, the admissibility condition requires H to be a lower bound estimate. For selecting shortest distance paths, the cost is measured in terms of actual distances; for minimum time paths, the cost is measured in time units (both distance between nodes and allowable velocity in the path segments are considered). The heuristic cost is calculated by assuming a straight line path having the maximum allowable velocity exists from the node in question to the goal node. An A* search procedure (with lower bound estimates) could be summarized as follows:

1. Form a queue of partial paths. Let the initial queue consist of the zero-length, zero-step path from the root node to nowhere.

2. Until the queue is empty or the goal has been reached, determine if the first path in the queue reaches the goal node.

   a. If the first path reaches the goal node, do nothing.

[20] N. Nilsson; P.H. Winston.

b. If the first path does not reach the goal node:

   (1) Remove the first path from the queue.

   (2) Form new paths from the removed path by extending one step.

   (3) Add the new paths to the queue.

   (4) Sort the queue by the sum of cost accumulated so far and a lower bound estimate of the cost remaining, with least cost paths in front.

   (5) If two or more paths reach a common node, delete all paths except the one that reaches the common node with the minimum cost.

An example of the path planning algorithm using A* search is provided in Appendix C.

Enhancements to the Control Algorithm

In this phase of research, the control algorithm was extended to deal with more than two robots. The search time increases with the number of robots. Constraints in the initial control algorithm were relaxed to allow obstacles to be placed in highways and to allow continuous scanning while the robots are moving between nodes. When obstacles are scanned in the path of the robots, the sensor provides information that is used to make navigation decisions (bypass or retrace).

Navigation planning is an important phase of the warehousing activity, especially when many robots are used for material transfer. Successful navigation requires coordination between various activities. The activities may include world representation schemes, prioritization of tasks based on demand, optimum path selection, collision avoidance, obstacle tracing and avoidance, and dynamic planning.

World representation has been achieved by choosing a network model of the warehouse layout. Warehouses usually have several blocks or modules and aisleways within each block where the robots perform pick/place operations. Also, there are alleys or short path segments connecting the aisleways. The aisleways and alleys are classified as pathways. In the network model, the pathways form the links or arcs, and the junctions or corners in the pathways form the nodes. Travel time in the aisleways is greater than in the other links.

It is difficult to predict the arrival of orders in a warehouse. However, they could be assumed to follow a random distribution pattern (such as Poisson distribution) or a uniform distribution pattern. Requests following a random distribution should be considered for planning. The appearance of a high priority order may cause radical changes in the plans of the otherwise 'busy' robots.

When alternate paths are available, the 'best' path must be selected. Since the robot travel velocities and the allowable pathway velocities can be altered, unloaded robots may be programmed to travel at their maximum speeds and choose the higher velocity pathways. The loaded robots, on the other hand may travel slower, and choose the shortest distance pathway.

Path planning is followed by execution of the collision avoidance algorithm. Since two or more robots may have to travel along the same path or may have certain common nodes in their paths, there is a need to set priority levels for each robot. The higher priority robot is allowed to move first while the lower priority robots wait in a queue. If the path of two robots contains common nodes, the robots may arrive at a node at the same time and collide. Therefore, it is important to set the arrival time at the nodes in the path for each robot. This activity creates a node-time chart for each robot that will ensure no collisions between the robots in the course of their travel based on their present direction of movement.

Collision avoidance in multiple robot environments is important for the system to function smoothly. Collisions may be prevented by stopping all robots in the vicinity of the colliding zone and allowing robots to proceed one at a time. Avoiding interaction could also be achieved by predicting possible collisions and altering the robot velocities or planning alternate paths.

The flexibility of the navigation planning system is governed by the ease with which paths can be altered without significantly affecting the optimization rules. If a robot finds an obstacle in its path, it can either back off to a new location or bypass the obstacle. The decisionmaking modules need to closely interact with the sensor modules and the navigation planning module.

To generalize the navigation algorithms, the path planning algorithm was enhanced to accept arbitrarily placed nodes. An example of navigation planning is provided in Appendix C.

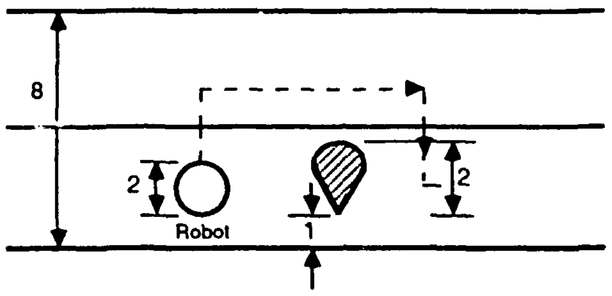
The path planning algorithm now executes the robot's movement from one node to the next based on the arrival times at the nodes. Movement is done successively by halting at a node, scanning in the direction of the next node in the path, and moving to the next node. If an obstacle is traced, the algorithm updates the network structure by removing branches that connect the robot's current node with the next node. New paths are searched for all robots from their respective nodes to the goals. The collision avoidance algorithm is called into operation and the whole process is repeated until the goal positions are reached.

"Dynamic" obstacles can appear when a robot drops an item or breaks down in the pathway. Therefore, the algorithm should be capable of making a decision at any point in its pathway. Also, the pattern of executing motion (stopping, scanning, and moving) does not allow continuous and smooth motion. With these factors in mind, the bypassing algorithm discussed below was developed.

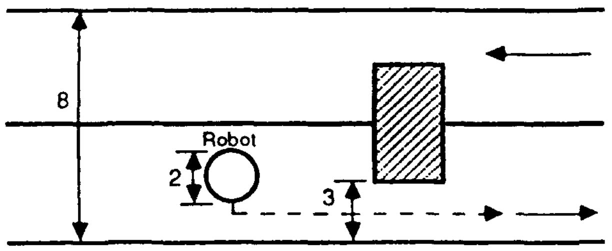
Bypassing Algorithm

This algorithm uses sensor data about an obstacle that a robot encounters. The data includes the obstacle's position with respect to the robot, the width of the obstacle, and the distance between the obstacle's position and the right side of the lane. The width of the obstacle is the width that is projected on a plane perpendicular to the sensing direction. The data is used to determine the possibility of bypassing the obstacle.

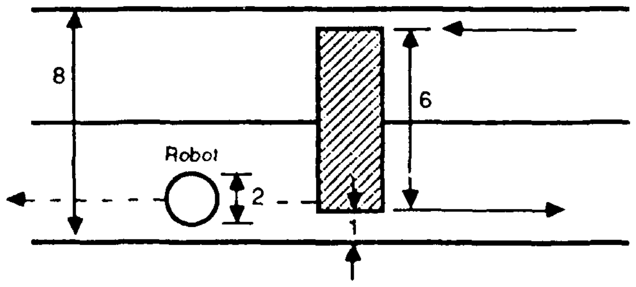
Figure 6 shows three situations where the bypassing algorithm is used. The robot is assumed to have a width of two units. The lane width is four units. The arrows indicate the direction of movement. Figure 6(a) shows an obstacle two units wide with its edge located one unit from the lane side. The lane is declared blocked and the robot may bypass the obstacle by moving into the other lane and traveling until it is past the obstacle. In Figure 6(b), the obstacle edge is three units from the lane side. The robot may shift its orientation as shown by the dotted line and continue along the same lane. In Figure 6(c), the obstacle blocks both lanes. In this case, the robot has no option other than to retrace its path.



(a) Robot bypasses



(b) Robot moves along the same path



(c) Path blocked; robot retreats

Figure 6. Rules for bypassing.

Bypassing usually results in moving against the present direction of movement. This requires a modification in the general collision avoidance algorithm that computes the arrival time at a node. When two robots are moving along the same path (one behind the other) and the first robot encounters an obstacle which can be bypassed, that section of the path is considered blocked for all robots except the robot that is bypassing the obstacle. In such an instance, the second robot will find an alternate path. The basis for this rule is that bypassing is assumed to require sensor interaction and the process is bound to be time consuming since it involves frequent stopping and scanning. Traffic moving in the opposite lane can also be affected as the bypass is executed.

Consider the section of the warehouse shown in Figure 7. Let Robot 1 trace an obstacle while moving from B3 to B3B5 and let Robot 2 also be moving from B3 to B3B5 behind Robot 1. If the sensor data shows that the obstacle can be bypassed, Robot 1 moves to the other lane (indicated by B8 $\rightarrow$ B8B10) and moves opposite the normal direction toward B8. Robot 1 informs the central computer about the path blockage. The central computer will then update the tree structure to remove the branch between B3 and B3B5. At this point, a new path is searched for Robot 2 from its most recent node to its goal. Robot 2 will retrace from its current position by reversing and will follow a newly computed alternate path to the target location. Bypassing is advantageous in situations where the alternate path may be too long to travel and the bypassing robot has a higher priority.

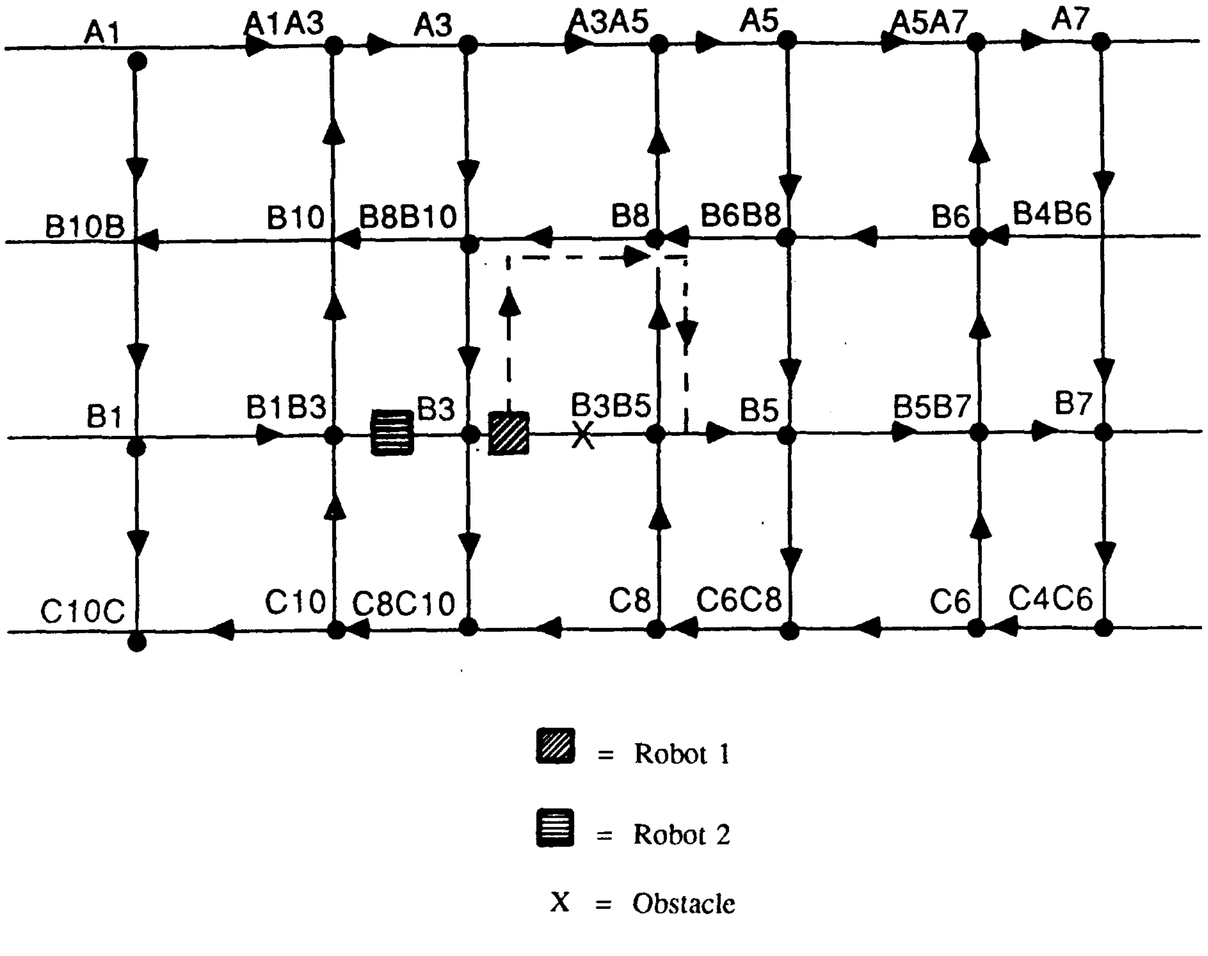


Figure 7. Network structure to demonstrate robot bypassing.

Since bypassing requires moving in the opposite lane, the question arises as to how far the robot can continue in that lane. The practical answer is to continue movement as long as the obstacle continues to be sensed in the other lane or until the robot encounters an obstacle in the lane of movement, in which case it may have to retrace its path. For the simulation, the bypassing robot always moves from its current position to the opposite lane and continues movement until it reaches the next node. It then moves back to the path it was following before the obstacle was sensed. In the context of the above example, Robot 1 moves from its current position to the B8 $\rightarrow$ B8B10 lane and moves towards node B8. After reaching B8, it moves to B3B5 and continues movement along the original path.

Collision Avoidance Algorithm for Bypassing

Bypassing calls for a change in the general collision avoidance routine. The previous strategy to avoid collision was to search for identical nodes in the paths of the robots, compare the arrival times, and alter the arrival times of the lower priority robots. This algorithm works when traffic is moving in the preset directions. The algorithm needs to be changed when a robot moves against the direction of normal traffic while bypassing. One solution is to stop all traffic heading toward the node to which the bypassing robot is moving until the movement is completed. This is obviously not efficient. To achieve a smooth flow of traffic, a look-ahead technique is necessary to foresee chances of collision with other robots and prevent them. With this fact in mind, the collision avoidance algorithm for bypassing was developed.

The collision avoidance algorithm for bypassing was based on the priority levels of the robots. The bypassing decision is immediately followed by a path search from the current positions of the robots (other than the bypassing robot) to their respective targets. The general collision avoidance routine is then called to compare identical nodes in the robots' paths and alter the arrival times if necessary. The collision avoidance algorithm for bypassing is then executed to control traffic, if any, moving along the bypassing lanes. To take care of any changes in the arrival times made in the collision avoidance algorithm for bypassing, the general collision avoidance algorithm is again executed. The various steps of the collision avoidance algorithm for bypassing are shown in Figure 8.

The following definitions are used in the flowchart.

BR = Robot that is carrying out the bypassing operations, also referred to as the bypassing robot.

N(BR,I) = Node through which the robot (BR) bypasses, also referred to as the bypass node. For example, in Figure 7, B8 is the bypass node.

N(BR,I) = Node following the bypass node in the path of the bypassing robot (e.g., node B3B5 in Figure 7).

N(BR,I-1) = Node preceding the bypass node in the path of the bypassing robot (e.g., the position of the bypassing robot when the obstacle was traced).

SUCC(BR,I) = Successor node of the bypass node that lies in the same segment of the path as that of the bypassing robot. In the nodal representation of the layout, the relationship between the nodes is obtained by establishing pointers for each node. The list of nodes that the pointer points to are called the successors of the node in question. For the example in Figure 7, B8 is the bypassing node in the path segment B8 $\rightarrow$ B8B10. The successors of B8 are B8B10 and A3A5 and therefore, SUCC(BR,I) is B8B10.

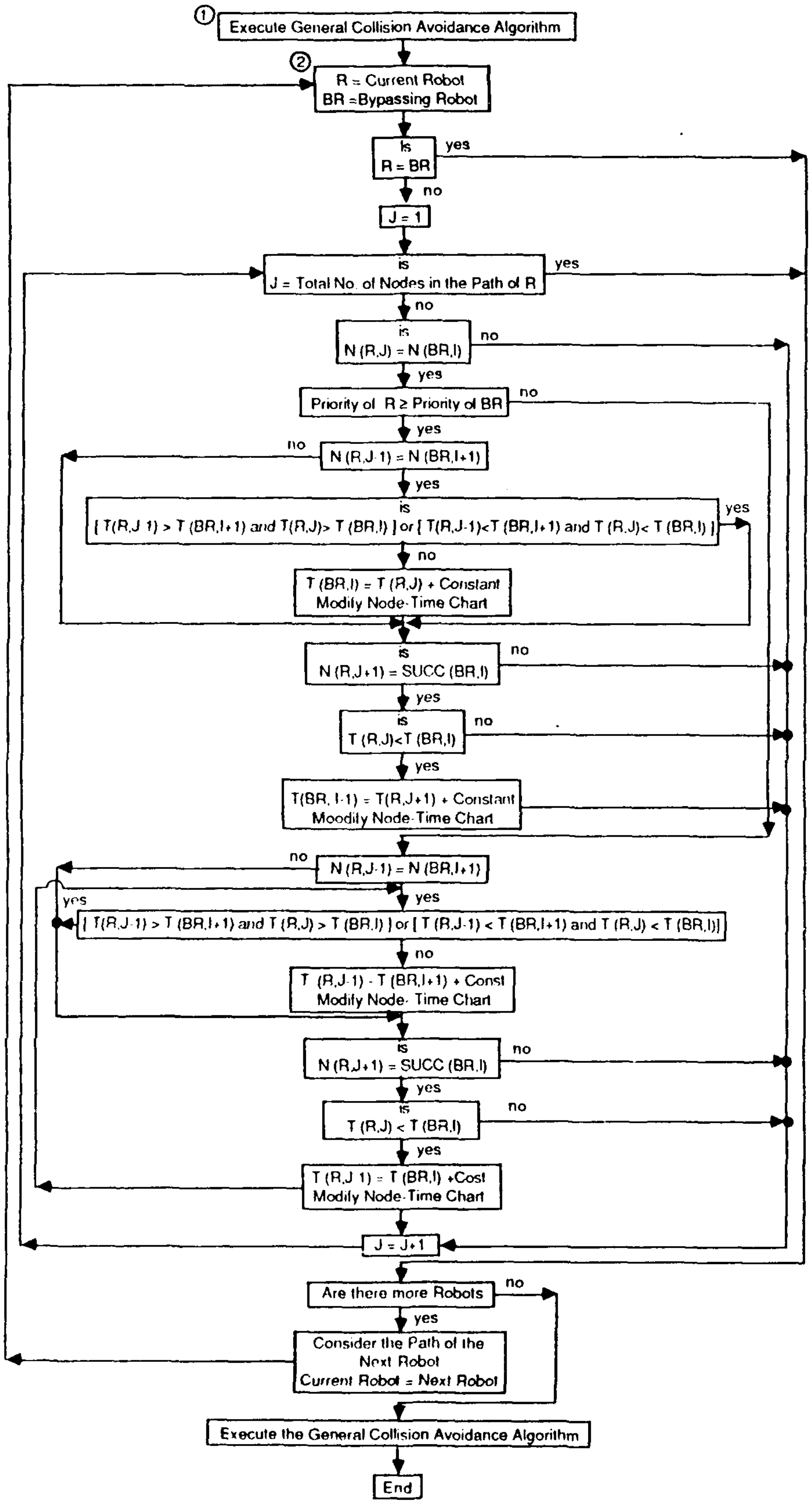

1
Execute General Collision Avoidance Algorithm
2
R = Current Robot
BR =Bypassing Robot
is R = BR
yes
no
J = 1
is J = Total No. of Nodes in the Path of R
yes
no
is N (R,J) = N (BR,I)
no
yes
Priority of R ≥ Priority of BR
no
yes
N (R,J-1) = N (BR,I+1)
no
yes
is [T(R,J-1) > T (BR,I+1) and T(R,J)> T (BR,I)] or [T(R,J-1)<T (BR,I+1) and T (R,J)< T (BR,I)]
yes
no
T (BR,I) = T (R,J) + Constant
Modify Node-Time Chart
is N (R,J+1) = SUCC (BR,I)
no
yes
is T (R,J)<T (BR,I)
no
yes
T(BR, I-1) = T(R,J+1) + Constant
Moodify Node-Time Chart
N (R,J-1) = N (BR,I+1)
no
yes
yes
[T(R,J-1) > T (BR,I+1) and T (R,J) > T (BR,I)] or [T (R,J-1) < T (BR,I+1) and T (R,J) < T (BR,I)]
no
T (R,J-1) - T (BR,I+1) + Const
Modify Node- Time Chart
is N (R,J+1) = SUCC (BR,I)
no
yes
is T (R,J) < T (BR,I)
no
yes
T (R,J-1) = T (BR,I) +Cost
Modify Node-Time Chart
J = J+1
Are there more Robots
no
yes
Consider the Path of the Next Robot
Current Robot = Next Robot
Execute the General Collision Avoidance Algorithm
End

Figure 8. Flow chart of collision avoidance algorithm for bypassing.

N(R,J) = the current node in the path of the robot under consideration.

T(R,J) = arrival time at node (R,J).

The structure of the bypassing collision avoidance algorithm is similar to the general collision avoidance algorithm. However, the rules set for altering the arrival times at the nodes are different. The general collision avoidance algorithm compares arrival times at a node and avoids collision by altering the arrival times. The collision avoidance algorithm for bypassing searches for the presence of the common bypassing node in the paths of the robots. Since the bypassing robot travels in a direction opposite to the preset direction, there is a chance of collision if any other robot travels in the path segments chosen by the bypassing robot. In Figure 7, the bypassing robot moves against the preset direction of movement in the path segments denoted by B8 $\rightarrow$ B8B10 and B3B5 $\rightarrow$B8. Hence, any robot that has these segments in its path is a candidate for collision due to bypassing. The bypass algorithm can be summarized in the following steps:

1. The decision to bypass results in an additional node (the bypass node) being added to the path of the bypassing robot. This results in a recomputation of the arrival times at the nodes for the bypassing robot. Any alterations in the node-time chart for one robot may cause collisions at other nodes in the paths of the robots. This situation is resolved by executing the general collision avoidance algorithm at the beginning of the bypass activity.

2. The priority of each robot is compared to that of the bypassing robot. If any robot priority is higher than that of the bypassing robot, the node arrival time of the bypassing robot is altered. On the other hand, if the priority of any robot is less than that of the bypassing robot, the node arrival time of the lower priority robot is altered.

3. For each robot path containing the bypass node, the nodes preceding and following the bypass node are compared to the nodes in the path segments of the bypassing robot. Rules mentioned in the flowchart are then applied to avoid collision.

4. The node arrival times are changed after comparing the path of the bypassing robot with each other robot. No effort is made at this stage to check if the changes in arrival times at the nodes (bypassing node, nodes following and preceding the bypass node) can cause collision at the other identical nodes that may be present in the path of the robots. Hence, the general condition avoidance algorithm is evoked at the end of the bypass algorithm.

The collision avoidance algorithm for bypassing is invoked only when bypassing is performed. It is always preceded and followed by the general collision avoidance algorithm. In collision avoidance for bypassing, the bypassing robot is penalized when higher priority robots are involved. Because bypassing is a slow and complicated operation involving frequent stops and sensor interaction, it is performed last. A detailed example of the collision avoidance algorithm for bypassing is presented in Appendix D.

5 WAREHOUSE SIMULATION

An animated color graphic simulation model was developed to demonstrate the capabilities of the algorithms that run the warehouse activities. The simulation model provides the user with a clear picture of how the control algorithms work. The user can study the layout of the modules and roadways, and the robots' movement governed by the algorithms and directed by the user.

Figure 9 shows a flowchart of the simulation, which consists of three phases. In the input phase, the user indicates the number of robots, their starting points, and destinations. The system starts by displaying the simulated layout of the warehouse (Figure 10). This layout forms a segment of the total warehouse and contains modules, highways, mainways, alleys, and junctions. Other segments can be added on the top or bottom of this segment. The boxes with alphanumeric symbols are the modules in which items are stored. A10 represents module 10 of group A. Similarly, D9 represents module 9 of group D. Rectangles A, B, C, D, AA, BB, CC, and DD are the junctions of mainways and highways. The outer two lines passing through the junctions represent the highway and enclose the segment. The horizontal lines in the segment are mainways and the vertical lines are alleys. Robots move in the direction indicated by the arrows. On the monitor, the modules are shown as colored rectangles. Junctions are also shown as rectangles, but are different size and color. The roadways are also shown with colored lines.

The letters and numbers shown on modules and junctions are identification names used to define the starting and destination points. Figure 11 shows an enlarged section of Figure 10. Alleys and bypasses are defined with respect to modules. For example, in Figure 11, bypass A1 is defined as the roadway in front of module A1. The short road section connecting the A1 and A3 bypasses is defined as bypass A1A3. The alleys are defined by the direction of motion and module. Alley B1 is defined by the vertical path going downward on the left of module B1 and alley C10 is defined by the vertical path going upward on the right of module C10. The small vertical path going upward and connecting the right side of modules A3 and A8 is alley A3A5. The path going downward connecting the left side is alley A8A10.

Each intersection of a mainway and an alley forms four nodes. These nodes are named depending upon their positions. For example, in Figure 7 an intersection contains nodes C1C3, C3, C8C10, and C10. The robots are shown as colored rectangles. A number written in each rectangle identifies the robot.

The system prompts the user for input and asks for the number of robots in use and their respective destinations. Due to system memory and processing time, the graphic simulation is limited to five robots.

The second phase, the static simulation, allows the user to introduce static constraints. Static constraints are defined as the known obstructions on the roadways (e.g., broken-down robots or fallen pallets). Constraints limit opportunities in path planning. Figure 12 shows an example of path planning with a static constraint. Node B1 is assigned as the starting node for Robot 1 and B9 as the destination node. The best path between B1 and B9 is given as:

B1 → B1B3 → B3 → B3B5 → B5 → B5B7 → B7 → B7B9 → B9.

If a static constraint is introduced in bypass B3 (as shown by an X), the next best path becomes:

B1 → B1B3 → C8C10 → C3 → C3C5 → C5 → C5C7 → C7 → C7C4 → C4 →B7B9 → B9.

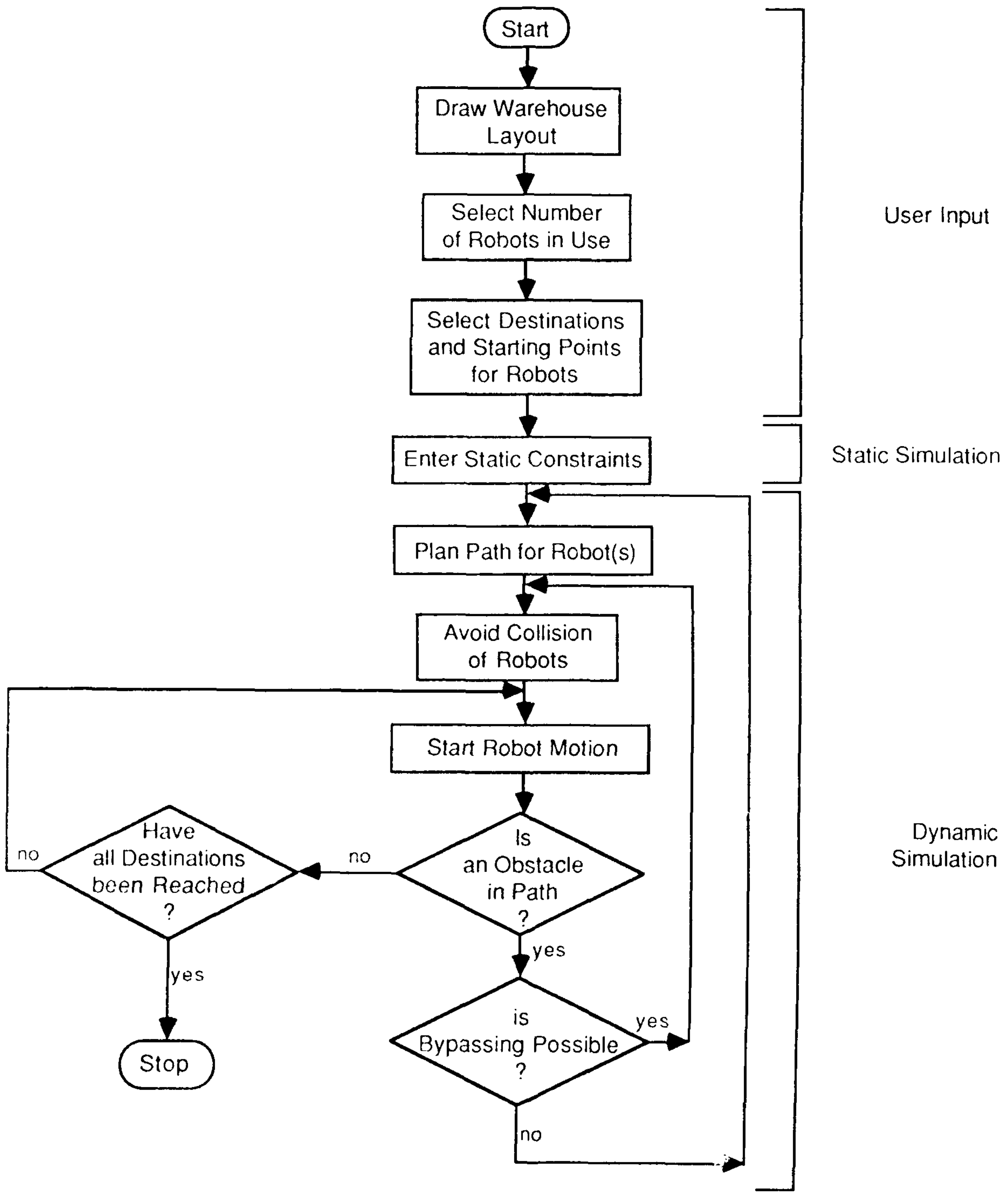


Figure 9. System flowchart.

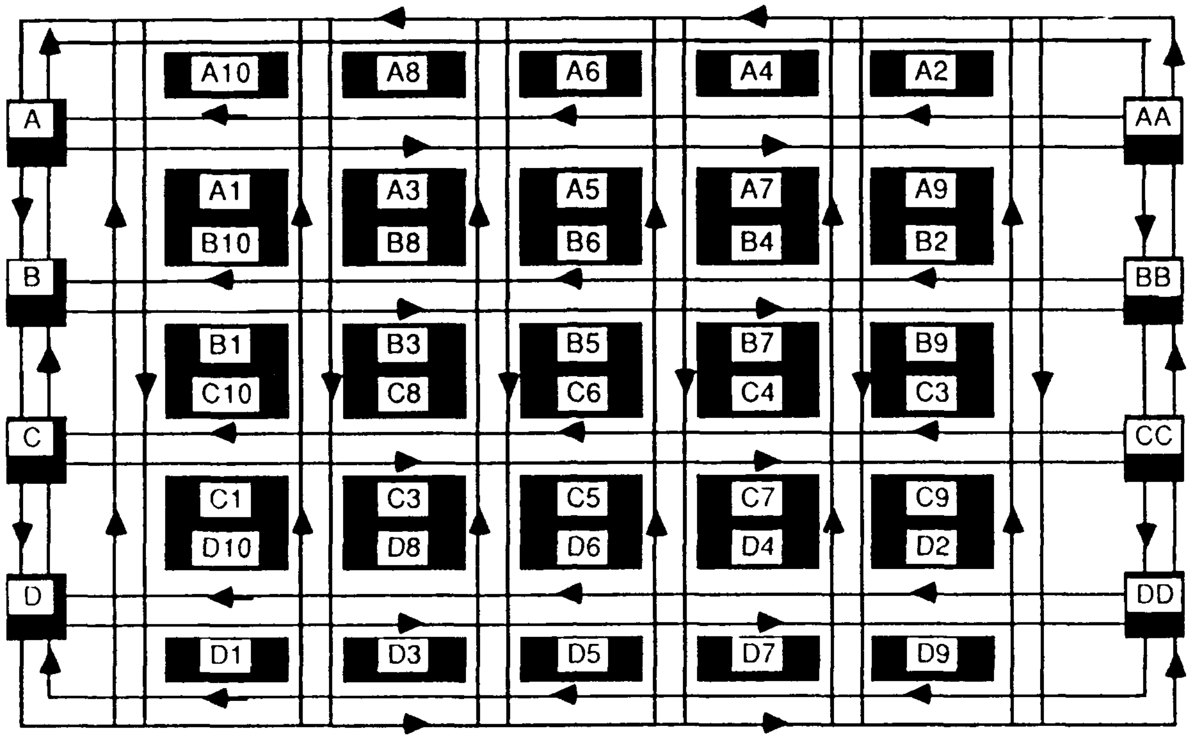


Figure 10. Simulated layout.

The third phase, the dynamic simulation, is the most critical. In this phase, robot motion is computed and shown on the monitor. Since robot motion is assumed to be priority dependent, the simulation assumes the highest priority for Robot 1, the next priority for Robot 2, and so on. Before the robots start to move, the collision avoidance algorithm sets their velocities and arrival times at different nodes so a collision will not occur. Figure 13 shows an example of collision avoidance. Robots 1 and 2 have the same path (B1 to B9). Since Robot 1 has higher priority, it starts first and Robot 2 follows at a safe distance.

The system allows the user to place obstacles in the path of moving robots by pressing the <RETURN> key. The system asks the user for the obstacle size and its location on the path. In actual use, the robots gather this information using sensors. Once the information about the obstacle is obtained, the bypassing algorithm decides if the robot can maintain its current path, if it can bypass the obstacle using the other lane, or if it must retrace its path because the obstacle is blocking the path. The new path for the robot is selected and motion is continued.

Figure 14 shows an example of bypassing the obstacle using the following path:

B1 → B1B3 → B3 → B8 → B3B5 → B5 → B5B7 → B7 → B7B9 → B9

Figure 15 shows another example of dynamic simulation. The path is completely blocked by obstacles in bypasses B3 and B8. The new path becomes:

B3 → C8C10 → C3 → C3C5 → C5 → C5C7 → C7 → C7C9 → C4 →B7B9 → B9

When all robots reach their respective destinations, the system will inquire if the user wants another run. If another run is not required, the program exits to the Disk Operating System (DOS).

The program was developed in IBM Advanced BASIC due to availability of graphics commands. The software is interactive and user friendly. The software is installed on IBM PC/AT-370 microcomputer with 640k RAM. The instructions to run the system are provided in Appendix E.

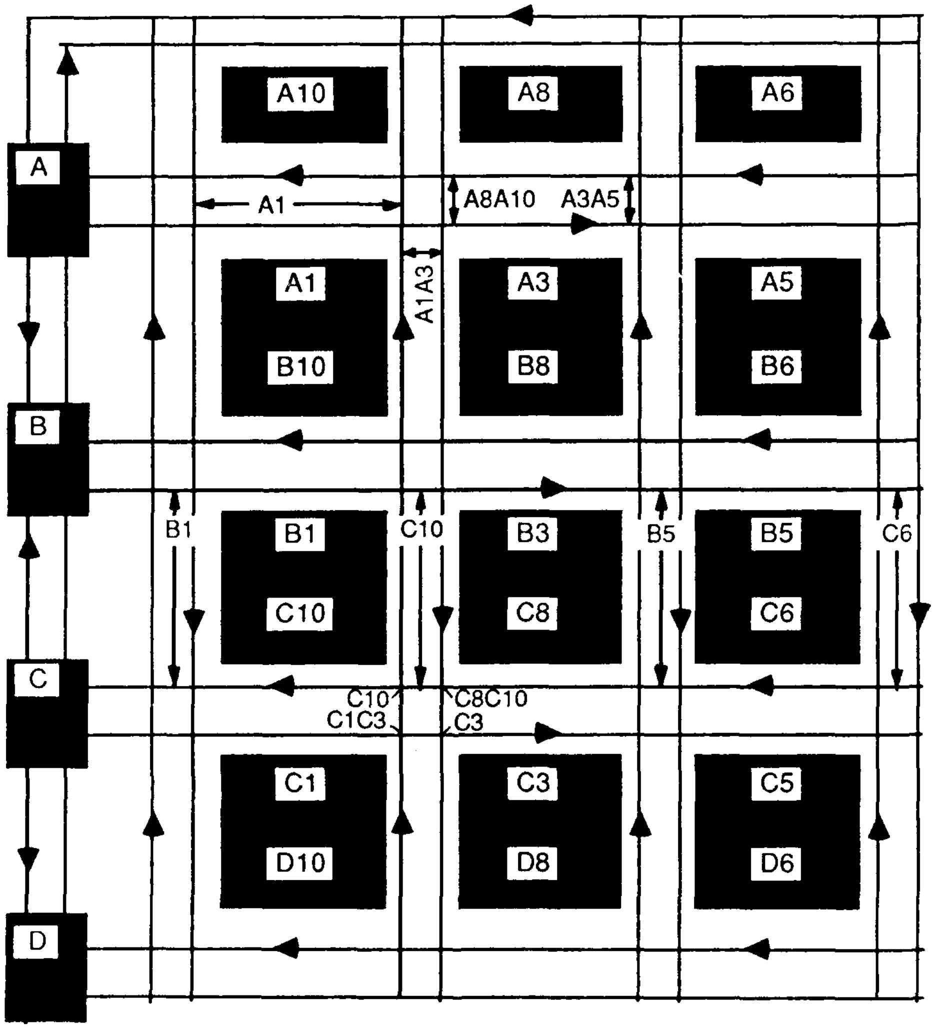
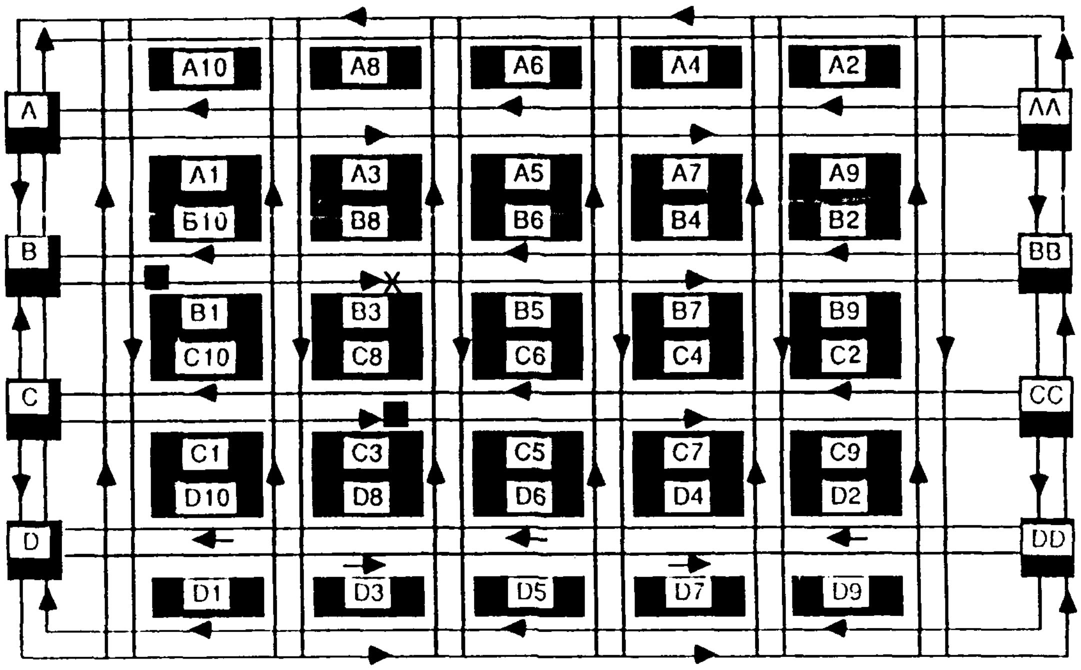


Figure 11. Identification of a bypass and an alley.



■ = Robot X = Constraint

Figure 12. Static simulation.

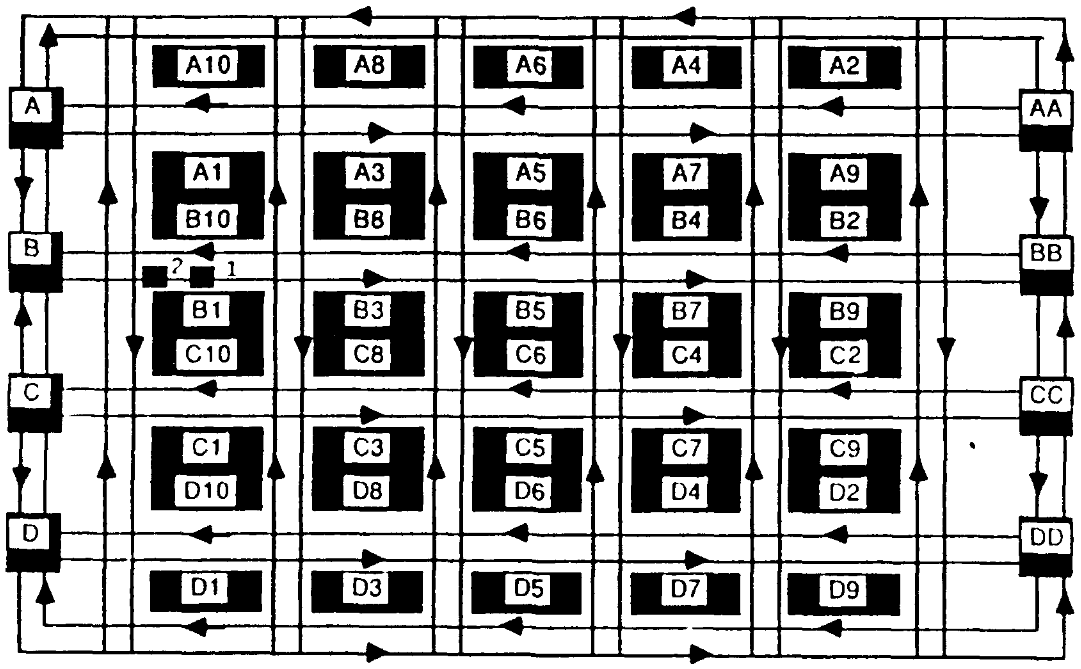
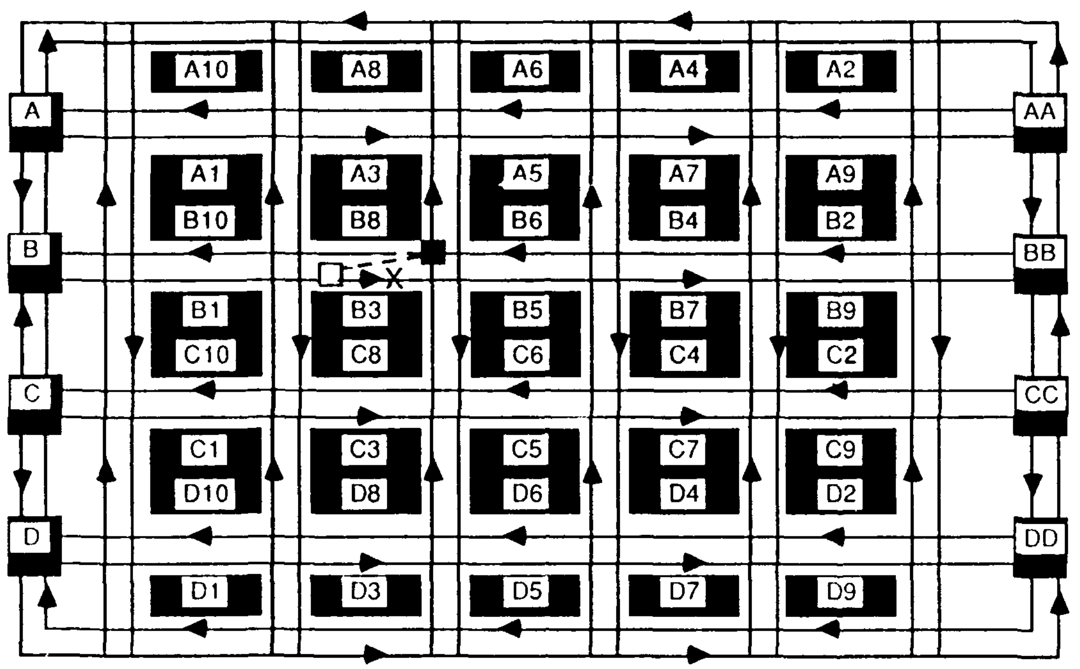
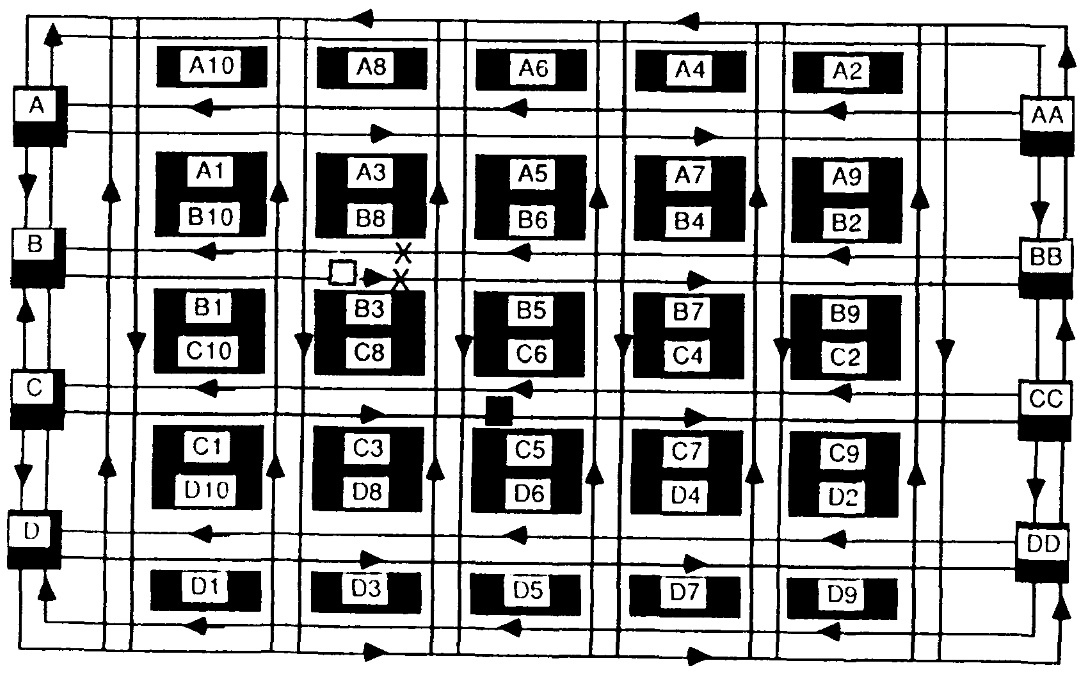


Figure 13. Dynamic simulation.



■ = Robot □ = Starting point of robot X = Constraint

Figure 14. Bypassing during dynamic simulation.



■ = Robot □ = Starting point of robot X = Constraints

Figure 15. Dynamic simulation of Robot 1 taking alternate path.

6 CONTROL AND COMMUNICATION ARCHITECTURES

In automated warehouses using autonomous mobile robots, a system's sophistication can be judged by its ability to make fast and intelligent decisions. Establishing control and communication architectures requires arranging the decisionmaking processors and allocating tasks to them such that the system is capable of making real time decisions. Hence, before establishing any architecture, it is essential to have a comprehensive overview of different tasks and how they affect the system.

Taking a broad view, the decisionmaking tasks of the system can be classified as decisions that affect the system globally or decisions which are local to individual robots. These decisionmaking tasks, though classified separately, are rigidly connected to each other. The decisions made at local level may significantly influence the decisions made at global level. It is essential to have communication links between processors at the global level and local level.

Control Architecture

Hierarchical distributed control architecture is the most suitable control system for the warehouse system. In this arrangement, the processors, or central processing units (CPUs) are arranged in levels as shown in Figure 16. The processors at any level can "talk" to processors in levels immediately above and below them. The processors at higher levels become the coordinators and supervisors for the processors at lower levels. Each processor can thus work independently, only consulting with its immediate superior, and does not get burdened with unnecessary information.

Two levels of processors are required in this application: one mainframe computer for the global level and robot on-board processors forming the local level. Allocation of tasks between these two processors is shown in Figure 17. The central computer governs and coordinates the movement of all the robots at a global level. Path planning, collision avoidance policies, the node-time chart data base, layout representation, and the status chart of each robot are accomplished by the central computer. The local level (i.e., on-board) processors are equipped with navigation control policies, sensor data interpretation, and integration algorithms.

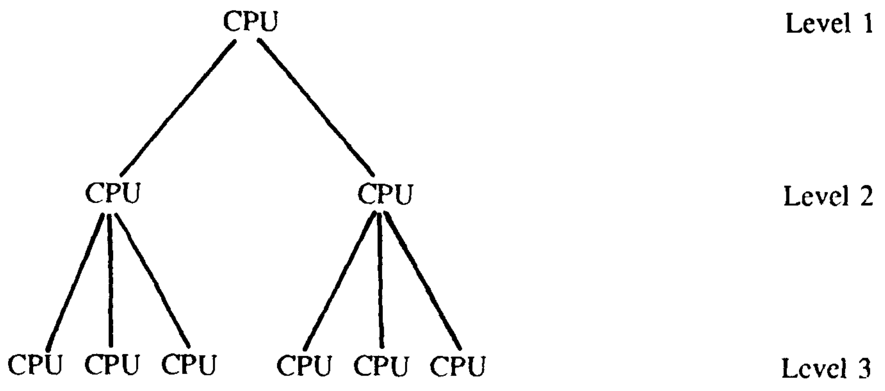


Figure 16. Processor arrangement for hierarchical control architecture.

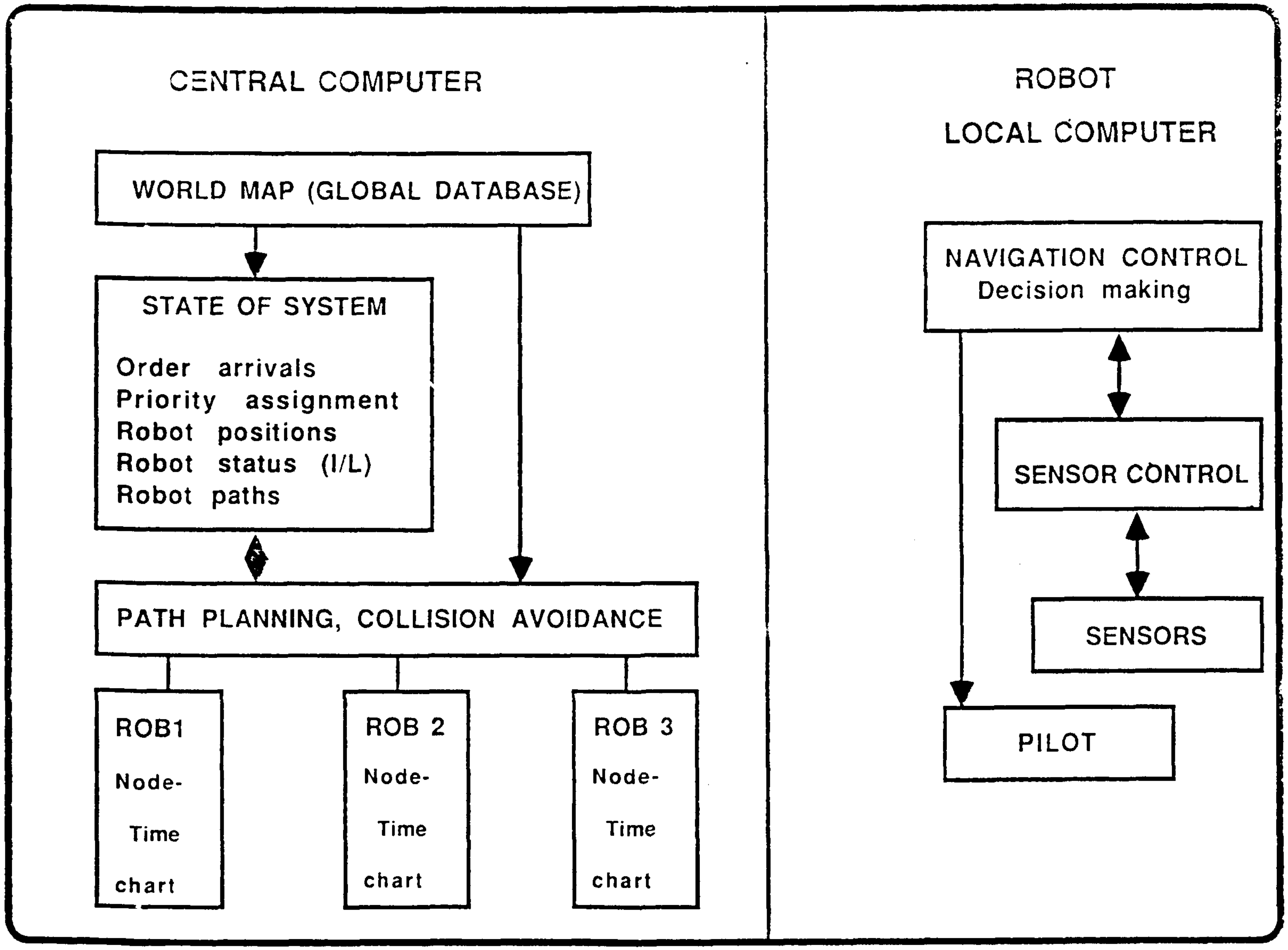


Figure 17. Allocation of tasks to robots and the central computer.

For example, suppose calls arrive at the central computer for placement of item A and retrieval of item B. Call A has priority 1 and call B has priority 2. The central computer assigns Robot 1 to handle call A and assigns Robot 2 to handle call B. The central computer retrieves the locations of items A and B from the data base and plans the paths for both robots. If the robots are arriving at any node at the same time, the central computer updates the velocities of the robots to avoid collision. The central computer communicates the respective planned paths and node-time tables to the robots. Figure 18 shows an example of exchange of information between central and local computers.

Communication Architecture

Because the communication link between robots and the central computer is essential to successful operation of the system, a robust link is needed. Communication systems can generally be grouped into two modes: wire communication and wireless communication.

In material handling applications, most AGVs use wire communication. Although this mode of communication is very robust and comparatively inexpensive, it is not very flexible. More new systems using wireless communication are being produced.

In wireless communication, radio and infrared systems are the most popular. For a warehouse, infrared communication systems are the most suitable because as the number of robots increases, radio communication develops problems of interference and noise. Also, radio support equipment is expensive. Infrared communication systems are very robust and the equipment is relatively inexpensive.

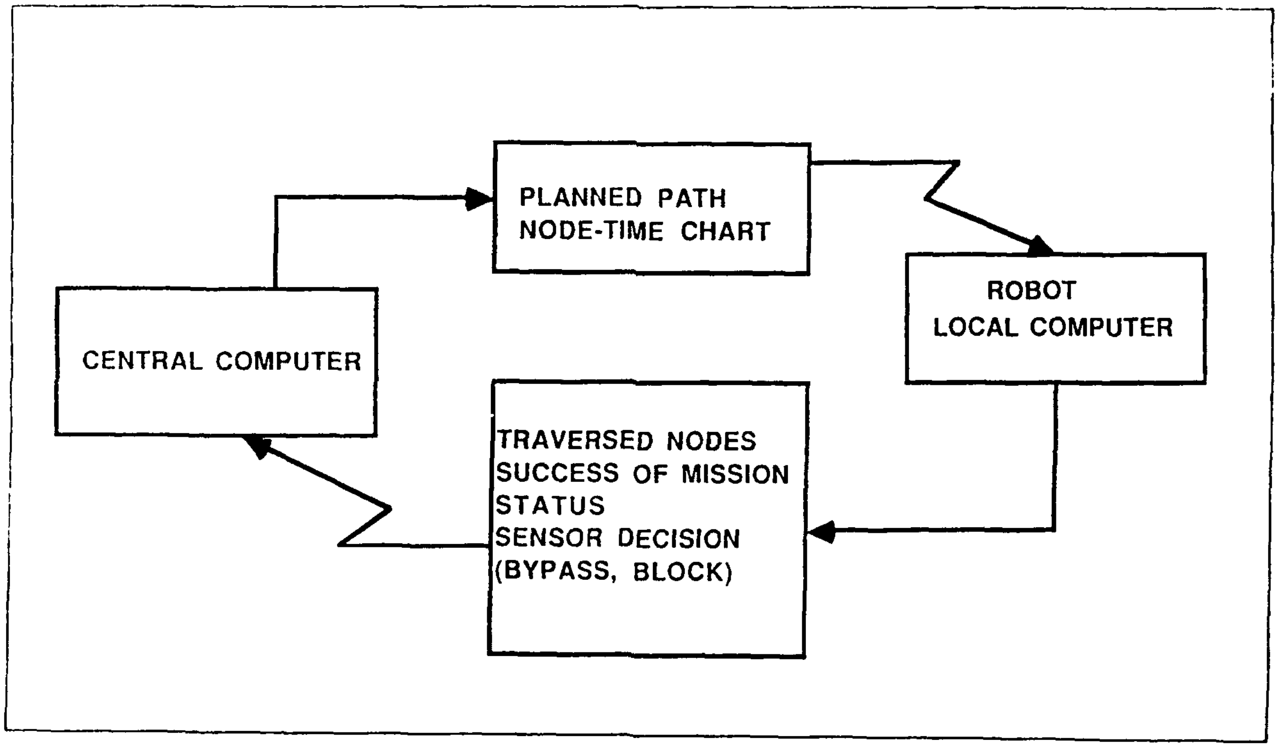


Figure 18. Communication links for information exchange.

Optodata 5200 (a product of Scientific Technologies, Inc. [STI]) was selected as the prototype communication system. It is a cableless optical data transmission device. This line-of-sight system provides one- or two-way transmission over moderate distances without the use of cable links, microwaves, or radio devices. It can also communicate data through high electrical noise areas or areas where cable connectors are virtually impossible because of potentially dangerous environments. It provides a low cost communication path between the central and local computers. The modulated infrared light carries asynchronous data at speeds up to 9600 baud and distances of 600 ft (183 m). Longer ranges are possible by using the unit as a repeater. The units are housed in rugged aluminum extrusions, sealed against moisture and dust. The remote power supply houses the data communication point. The input voltages are 100 to 130 volts alternating current (VAC), 200 to 260 VAC (50 to 60 hertz) or 24 to 48 volts direct current (VDC). Standard inputs available are RS-232C, RS-422, and 8-bit parallel. Two full sets of duplex systems may be used side by side without interference because each transmitter/receiver pair can be modulated on one of four different sets of frequencies. The proposed communication should be integrated into the control system using two layers of CPUs to evaluate response time, transfer rates, and reliability.

7 SUMMARY

In this phase of work, a user-friendly preprocessor was created to input the information about warehouse layouts. The preprocessor offers flexibility in defining and placing blocks and defining pathways and junction nodes. The output is a nodal data base containing world coordinates for the nodes, successors for each node, and link velocities. A second preprocessor was created to study layout design using data regarding the transfer of items and pallet dimensions. The preprocessor can handle layouts up to 200 nodes.

The control algorithms for navigation were enhanced to plan paths that consider optimizing criteria for loaded and unloaded robots and to use the heuristic A* search technique.

The updated control algorithm discussed in Chapter 4 can accept sensor input from more than two robots and can make navigation decisions based on that input. This interaction allows robots to bypass obstacles or follow new paths to their goal. A newly developed algorithm prevents collision specifically during the bypass process.

The sensors for navigation were identified based on a market survey on the available sensors. The 'best' systems were identified for use in future work. Visual sensors are used for pattern identification, scene analysis, and ranging. Sonar range finders, contact switches, magnetic sensors, goal-seeking sensors, and infrared sensors are used as proximity sensors and help the robots avoid obstacles. Wheel encoders are used to estimate the position/location of robots. Laser range finders are used to determine long distances. Gyroscopic level sensors aid robot navigation by providing information about the steepness of a slope. Robots can use tactile sensors to determine the shape of obstacles.

The control architecture of the system was set up as a hierarchical structure. The communication links between the central computer and individual robots were identified.

The color graphic simulation model runs on an IBM PC/AT-370 and was updated to allow action of more than two robots and to accept sensor data. The user inputs the number of active robots, their starting points, and their destinations. The user can also input static constraints (closed or blocked roadways) before the robots begin to move. The dynamic simulation allows the user to place obstacles in the path of moving robots. The simulation can be viewed on a graphical screen or a status screen. During the simulation, the user can use a tabular form to check the position or status of all robots.

8 RECOMMENDATIONS

Development Plan

Future research on autonomous warehouse systems using mobile robots must be directed toward integrating sensor and communication systems into the control system and physically demonstrating the capability of the integrated system to operate fully automated forklifts. The control system needs to be modified to accept input from the sensors and to interact with the communication package. This work can be accomplished at the branch level using a module platform and candidate sensor/communication systems.

After completing the bench tests and demonstration, the control systems should be mounted on forklifts and adjusted based on equipment response parameters. Modified forklifts could then be used to field test and demonstrate the capability of the control system. The completed system would be capable of moving forklifts throughout the warehouse, avoiding collisions, and finding the assigned target areas by using the optimum path. The integrated system can be interfaced with existing programs for controlling the pallet lifting, acquisition, and setting down processes, or extended in the future to handle these operations. Since the system is independent of floor tapes, it could be adapted for use in storage yards.

Sensors

The sensor system includes a variety of sensors. Integration of the sensors becomes important when two sensors supply different types of data. Work in this area should involve developing a test frame (mobile platform), monitoring the sensors and analyzing their performance, and integrating the sensor system into control architecture.

Depending on availability and compatibility, the following recommendations are made for the sensors needed in an automated warehouse:

- Position Referencing Sensors for Landmarks. Modulated infrared sensors, laser proximity sensors, or goal-seeking sensors are recommended for position referencing. Global positioning sensors and goal-seeking sensors using beacons are the least expensive. Although the laser proximity sensors give accurate information, they are expensive.

- Landmark Tracking Sensors. These sensors adjust the orientation of the robots with respect to the landmarks. They are recommended for use with bar code landmarks since the bar code readers have to be properly aligned with the cards containing the bar codes. Magnetic proximity sensors or infrared sensors could also be used.

- Landmark Decoders. Simple bar codes should be decoded using infrared readers because pattern analys equires a vision system.

- Remote Scanning. Long distance scanning is performed from each node toward the next node in the robot's path. Laser range finders are recommended for this purpose. Ultrasonic arrays could also be used.

• Short Distance Ranging for Obstacle Avoidance. Multielement ultrasonic arrays are recommended for short distance ranging. Adjusting the frequency of the sound pulses would enable an accurate estimation of the obstacle's position. In this use, rotating sensors are recommended to trace the obstacle's shape.

• Adjustment for Angular Movements. Angular movements should be calculated using gyroscopic sensors mounted on the chassis.

• Position Estimators. Wheel encoders are recommended for use in conjunction with the landmark information to determine global positions for the robots.

• Scene Analysis. Scene analysis is best performed using a vision system with solid state cameras. Ultrasonic arrays could also be used to some extent.

• Pick and Place Operations. An infrared proximity sensor and a magnetic elastic tactile sensor (for grasping) are recommended for loading and unloading activities.

Because the sensor technology depends on the warehouse environment and the mode of communication, robots should be able to interpret the sensor data and communicate with each other and with the central computer. Radio, infrared, and optic fiber links have been used with some mobile robots. It is recommended that problems associated with using these communication links in an open environment be researched.

Material Storage

Based on an extensive evaluation of the warehouse automation system carried out with the aid of the simulation package, research on the material storage policy should:

• Determine the optimum number of robots based on the size of the warehouse, demand patterns, failure frequency, and acceptable service times. Queuing and simulation models are recommended.

• Minimize traffic congestion to reduce chances of collision and failure. Consider zoning restrictions on robots and storing identical material in many locations to reduce traffic congestion. Zoning restrictions will involve partioning the warehouse into zones in which one or more robots will operate. The complexity of path planning algorithms will depend critically on the zoning policy used. Zoning may also affect material allocation policies.

• Resolve the problems of selecting routes for multiload vehicles. Develop and analyze operating policies for such a system.

Navigation Control

Based on the system evaluation, research on the operational (control) policy for navigation should:

• Minimize service time by optimal path planning and incorporate spatial and temporal aspects of the system in the path planning algorithms. Since routing and scheduling are done in real time, exact algorithms may not be feasible. Heuristics and rule-based systems should be developed to route the vehicles. Because the nodes may not be equidistant in most practical situations, an appropriate search technique will have to be developed.

- Use data from many sensors to build an abstract two-dimensional model of the environment containing the obstacle, and develop algorithms to interpret multisensor data and identify the obstacle. Initially the obstacles should be assumed to have a regular shape.

- Study local decisionmaking, as opposed to global updating and planning, in detail. The learning process should use efficiency and time as deciding factors for optimization. The global map should be updated with the position of a static obstacle and its configuration as gathered by sensors. The algorithm should then consider the obstacle while planning future paths. A knowledge base for the decisionmaking rules should be developed.

- Study enhancements to the collision avoidance strategy other than adjusting the robots' velocities.

- Develop a failure analysis system. The system should be capable of detecting failure, diagnosing causes, and suggesting appropriate remedial action. A knowledge-based expert system that uses analytic models of robotic subsystems should be developed for this purpose. The system should first identify if the failure is in the robotic system or in the environment. Failure in the robotic system should be analyzed using sensor data and data obtained by running tests on electrical, mechanical, and control subsystems of the robot. Failures in the environment (obliterated landmarks due to wear or dirt accumulations, or failure in the communication system) should be identified. The robot should be able to identify the kind of failure and signal for help.

- Develop and analyze simulation models for the above modifications.

CITED REFERENCES

Brooks, R.A., "Solving the Find-Path Problem by Good Representation of Free Space," *IEEE Transactions on System, Man and Cybernetics*, Vol SMC-13, No. 3 (Institute of Electrical and Electronics Engineers, 1983), pp 190-197.

Crowley, J.L., "Navigation for an Intelligent Mobile Robot," *IEEE Journal of Robotics and Automation* (1985), pp 31-41.

Gouzenes, L., "Strategies for Solving Collision-Free Trajectories Problem for Mobile and Manipulator Robots," *International Journal of Robotics Research*, Vol 3, No. 4 (1984), pp 51-64.

Imai, J., Y. Kawashima, and K. Hironaka, "Advanced Automated Transportation System," *Procedures of the First International Conference on Automation in Warehousing* (Mitsubishi Electric Corporation, Japan 1975), pp 175-184.

Jorgensen, C., W. Hamel, and C. Weisbin, "Autonomous Robot Navigation," *BYTE* (January 1986), pp 223-235.

Kapoor, S.G., et al., *Development of a Guidance System for Automated Warehouse Equipped with Mobile Robots*, Technical Report P-87/07 (U.S. Army Construction Engineering Research Laboratory [USACERL], February 1987).

Keirsey, D.M., et al., "Algorithm of Navigation for a Mobile Robot," *IEEE International Conference on Robotics and Automation* (1984), pp 574-583.

Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *IEEE International Conference on Robotics and Automation* (1985), pp 500-505.

Komonya, K., and K. Tamie, "Research Review for Realization of Autonomous Locomotion," *Materials Flow*, Vol 3 (1986), pp 3-15.

Krogh, B.H., and C.E. Thorpe, "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles," *IEEE International Conference on Robotics and Automation* (1986), pp 1664-1669.

Lasecki, R.R., "AGVs: The Latest in Material Handling Technology," *CIM Technology* (Winter 1986), pp 90-94.

Lozano-Perez, T., and M.A. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Communications of the ACM*, Vol 22, No. 19 (Association for Computing Machines, 1979), pp 560-570.

Lu, S.C-Y, T.P. Brand, and S.G. Kapoor, *Sensor and Guidance Technology Related to Mobile Robots*, Technical Report P-87/02 (USACERL, June 1987).

Moravec, H.P., "The CMU Rover," *Proceedings of the National Conference on Artificial Intelligence* (American Association of Artificial Intelligence, August 1980), pp 377-380.

Nilsson, N., *Principles of Artificial Intelligence* (Tioga Publishing Company, 1983).

Oliver, J.L., and F. Ozguner, "A Navigation Algorithm for an Intelligent Vehicle with a Laser-Range Finder," *IEEE International Conference on Robotics and Automation* (1986), pp 1145-1150.

Siy, P., "Road Map Production System for Intelligent Mobile Robots," *IEEE International Conference on Robotics and Automation* (1984), pp 562-570.

"The Fundamentals of Machine Vision System," *Vision Technology* (June 1986), pp 2-9.

Winston, P.H., *Artificial Intelligence* (Addison-Wesley, 1984).

UNCITED REFERENCES

Batta, R., and D. Malon, *The Mobile Server Problem* (Operations Research Society of America/The Institute of Management Sciences [ORSA/TIMS] Joint National Meeting, Miami, Oct. 1986).

Iyengar, S.S., et al., "Robot Navigation Algorithm Using Learned Spatial Graphs," *Robotica*, Vol 4, Part II (April-June 1986), pp 93-100.

Litton AGVs, brochure (Litton Industrial Automation Systems, Inc., Automated Vehicle Systems, Zeiland, Michigan 49464, 1986).

Pau, L.F., "Survey on Expert Systems for Fault Detection, Test Generation, and Maintenance," *Expert Systems*, Vol 3, No. 2 (April 1986), pp 100-111.

Thorpe, C.E., and L.H. Matthies, "Path Relaxation: Path Planning for a Mobile Robot," *IEEE Journal of Robotics Research* (1984), pp 576-581.

APPENDIX A:

PREPROCESSOR I

Step 1: User Input

The user initializes the system and defines the rack arrangements (blocks) used in the warehouse. The following inputs are needed from the user:

1. Maximum length of the warehouse,
2. Maximum width of the warehouse,
3. Total number of rack arrangements (blocks) used,
4. Dimensions of each block type,
5. Width of aisle associated with each block type, and
6. Width of highway.

The system scales the graphic screen to the specified warehouse length and width and creates each type of rack arrangement in horizontal and vertical orientations. This provides the user with an extended choice of arranging the blocks in vertical, horizontal, or mixed patterns.

Step 2: Arrangement of Blocks

The system starts with a blank layout scaled to warehouse dimensions. The user is prompted to make a selection from the available block types. As the user makes the selection, the block appears in one corner of the layout and the user can move it to the desired position using arrow keys. Once the block has been properly placed, the user can insert it in the layout using the "Ins" key. All the blocks in the layout are placed in this manner.

Step 3: Placement of Highway Junctions

The user can incorporate a highway in the layout. This step is optional because a highway may not exist in the layout. This step is skipped if the width of the highway is defined as zero in Step 1. If the step is executed, the user is provided with junction blocks scaled to the width of the highway. The user can move these junctions around the layout and insert them. Since the highway passes through these junction blocks, a highway of any desired shape can be created.

Step 4: Defining Pathways Between Landmark Nodes

Once all the rack arrangements and junction blocks are placed, the next step is to define pathways and the direction of motion. The user is provided with a cross-like cursor to locate nodes between which the pathways need to be created. Pathways inside each block are predefined in a clockwise direction. To define pathways between two landmark nodes of different blocks, each node is indicated by positioning the cross over them. Once both nodes are indicated, they are connected with a line indicating a pathway between them. The direction of motion in any such pathway is defined while indicating the node. The direction of motion is always from the first node mentioned to the second.

Step 5: Nodal Representation of Layout

Planning the movement in a particular direction requires the definition of specific points along the path. With this in mind, the global environment is represented as a work model. The nodes represent points where a change in the direction of traffic is expected. They also represent the landmark locations. Generating the nodal structure is important to the simulation. A scheme to automatically generate the nodal structure for any layout created is explained in the following paragraphs.

Each block placed in the layout is identified by a number. The nodes associated with this block are also numbered. The numbering is done as follows. Let the block number be i, then the nodes are numbered as N_{ij}, where:

$$N_{ij} = 4(i-1) + j \quad j = 1 \ldots 4.$$

The block numbers are shown on the screen but the node numbers are internally stored. The coordinate location of each block and its nodes are determined with respect to the layout boundary. As the block is moved around the layout in Step 2, its position is continuously updated with one of the boundary corner points as reference. Once the block is inserted, the coordinates of all the nodes are easily determined. For example, let the lower left hand corner of the block be made the reference point for that block (x,y). As the block is moved around, the layout point (x,y) is updated depending upon the direction of movement. Once the block is placed, the point (x,y) is known. It is assumed that robots always move in the middle of the lane. Hence, the nodes are located in the middle of the lane. Now the coordinates of the nodes are calculated. For example, the coordinates of node 1 will become:

$$X_1 = X + 0.5 * ASL$$

and

$$Y_1 = Y + 1.5 * ASL + 2*WR$$

where ASL = width of aisle
WR = width of rack.

The identification of each node, its coordinates, and the connecting nodes (defined in Step 4) form the nodal representation of the warehouse.

The nodal representation and the color graphic layout form the output of the preprocessor. An example of a layout (in black and white) is shown in Figure A1.

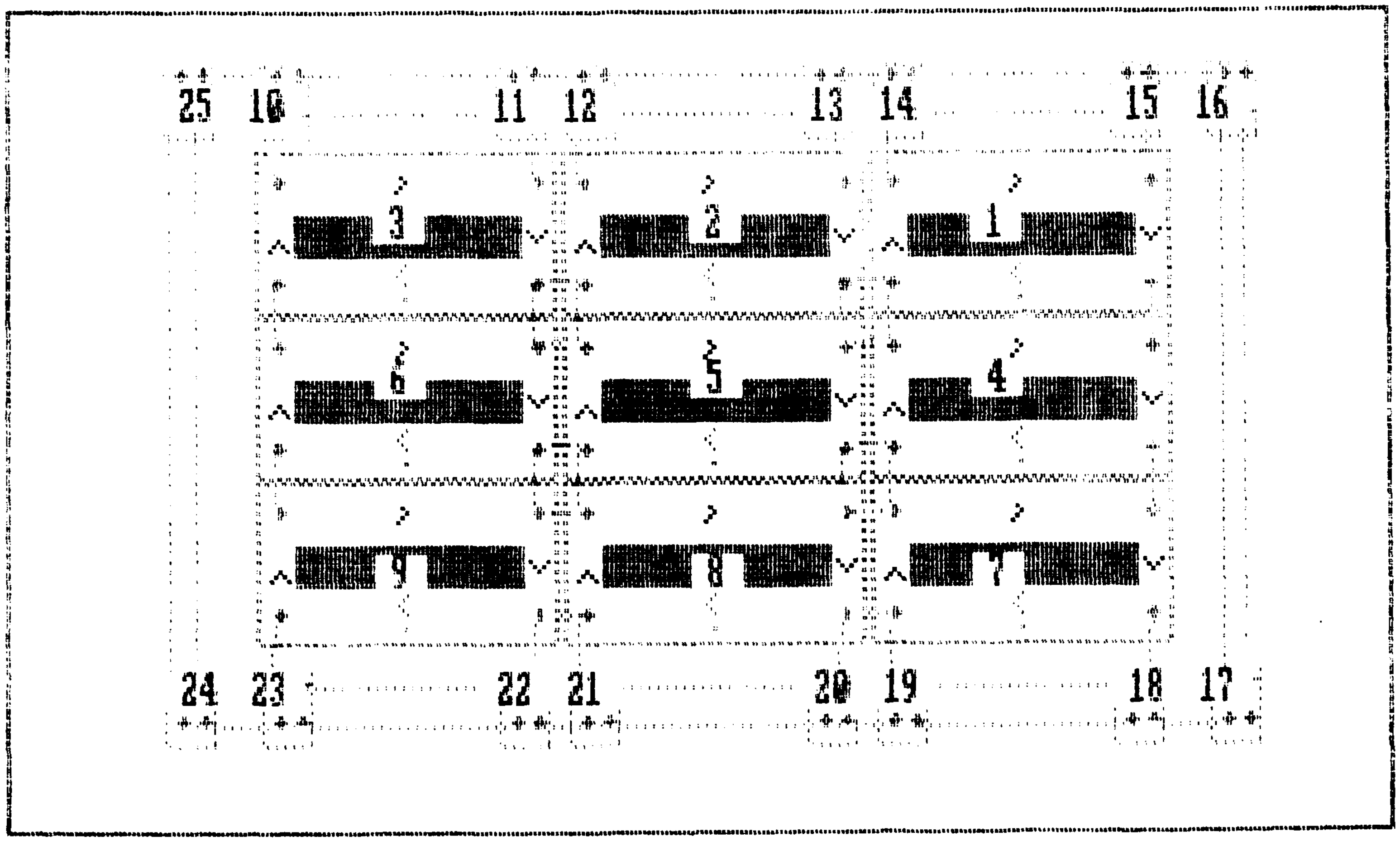


Figure A1. Layout generated by preprocessor I.

A sample nodal data base generated by preprocessor 1 for the designed layout (Figure A1) is shown in Figure A2. The data base lists the total number of nodes, total number of mainway nodes, and the mainway and highway velocities in the first line. This is followed by nodal data which consists of node number, X and Y coordinates of the node, the total number of successor nodes, the successor node and the velocity of the link joining the node to the successor node, and the second successor node and the velocity of the link.

For example, referring to Figure A2, the layout has 100 nodes, of which 36 are mainway nodes. The mainway velocity is 1 unit and highway velocity is 2 units. Node 1 has its coordinate at (200, 35), two successors; the first successor is node 2 and velocity in the link 1 $\rightarrow$ 2 is 1 unit. Second successor is node 56 with link (1 $\rightarrow$ 56) velocity of 1 unit.

The system also provides the user with a readymade layout consisting of mainways, alleys, and highways for the simulation of control policies. The nodal representation of this demonstration layout is already created and stored. This layout can be used to quickly demonstrate navigation policies.

Node #	X coord.	Y coord.	# of successor	Successor node	Velocity	2nd succ. node	Velocity
100	36	1	2				
1	200	35	2	2	1	56	1
2	260	35	1	3	1		
3	260	55	2	4	1	14	1
4	200	55	2	1	1	7	1
5	130	35	2	6	1	48	1
6	139	35	2	7	1	1	1
7	189	55	2	8	1	18	1
8	130	55	2	5	1	11	1
9	60	35	2	10	1	40	1
10	120	35	2	11	1	9	1
11	120	55	2	12	1	12	1
12	60	55	1	9	1		
13	200	68	2	14	1	4	1
14	260	68	1	15	1		
15	260	88	1	26	1		
16	200	88	2	13	1	19	1
17	130	68	2	18	1	8	1
18	189	68	2	19	1	13	1
19	189	88	2	20	1	30	1
20	130	88	2	17	1	23	1
21	60	68	2	22	1	12	1
22	120	68	2	23	1	17	1
23	120	88	2	24	1	34	1
24	60	88	1	[illegible]	1		
25	200	102	2	26	1	16	1
26	260	102	1	27	1		
27	260	123	2	28	1	70	1
28	200	123	2	25	1	31	1
29	130	102	2	30	1	20	1
30	189	102	2	31	1	25	1
31	189	123	2	32	1	78	1
32	130	123	2	29	1	35	1
33	60	102	2	34	1	24	1
34	120	102	2	35	1	29	1
35	120	123	2	36	1	86	1
36	60	123	1	33	1		
37	60	13	1	38	2		
38	65	13	2	39	2	41	2
39	65	21	1	40	2		
40	60	21	2	37	2	97	2
41	114	13	1	42	2		

Figure A2. Partial nodal data base generated by preprocessor I.

APPENDIX B:

PREPROCESSOR II

The following terms are defined to simplify the discussion on warehouse design.

Grid: Storage space in which material is stored. A grid cannot be deeper than the horizontal reach of the robot.

Rack: A set of grids is defined as a rack. A rack cannot be higher than the vertical reach of the robot.

Block: A block consists of 2n of racks with n racks on each side. Racks are placed back to back. The 2n racks are surrounded by an aisleway on four sides as shown in Figure 2 of the main text.

In Figure B1 a grid, rack, and block are shown in detail. To design the physical system, the following assumptions were made:

1. Materials are stored on pallets,
2. Each type of material has a fixed pallet size,
3. Each grid contains only one type of material,
4. Pallets are stored in the same orientation as they are transported on the forklift, and
5. All blocks are of the same size and all racks are of the same size.

length of a block = LB = NRACK + LR + 2*WSA

width of a block = WB = 2* WR + 2* WFA

Area required by a block

BAREA = LB x WB

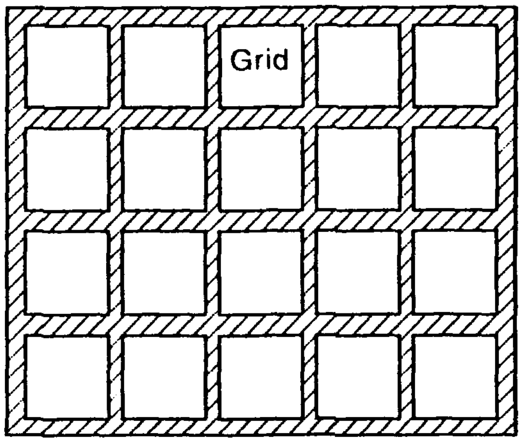
Total area required

AREA = BNUM * LB * WB

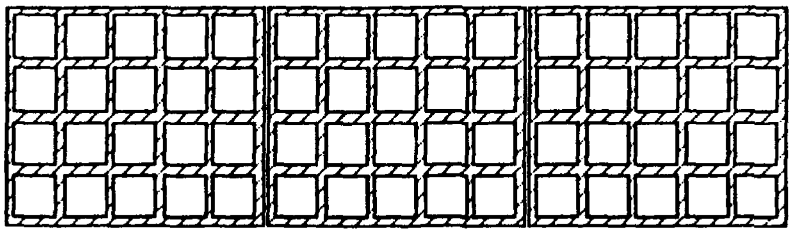
The following variables are not defined in the main text but are used to design the warehouse:

NUM	= number of material types to be stored
PP_i	= number of pallets of material type i to be stored i = 1.,,, NUM.
LP_i	= length of pallet of material type i
WP_i	= width of pallet of material type i
HP_i	= height of pallet of material type i
LG	= length of grid
WG	= width of grid

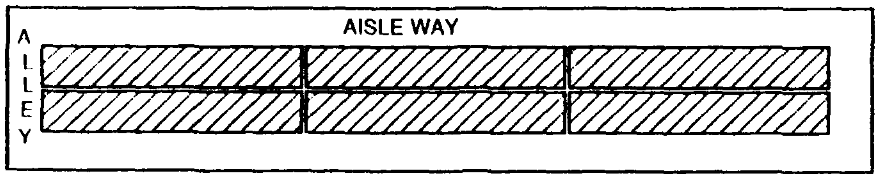
HG	= height of grid
CL	= clearance between two grid spaces
NR	= number of grids in one row of rack
NC	= number of grids in one column of rack
LR	= length of rack
WR	= width of rack
HR	= height of rack
NRACK	= number of racks on one side of a block
WSA	= width of side aisle
WFA	= width of front aisle



Storage Rack



Front-View of a Block



Top-View of a Block

Figure B1. Grid, rack, and block.

Arrangement of Blocks: (Layout Planning)

The blocks are arranged such that the layout will be nearly square. Since the blocks formed are rectangular, the final layout is rectangular. The procedure to arrange blocks is as follows.

Step 1: Assuming the required space (AREA) is a square, find the length and width of the square space (LS and WS, respectively):

Length of square = LS = $(\text{AREA})^{1/2}$
Width of square = WS = $(\text{AREA})^{1/2}$

Estimation of Required Number of Blocks

The number of racks, and hence the number of blocks needed to store all the pallets can be estimated as follows:

number of pallets in one length of a grid
$AL_i = LG/(LP_i + 2*CLER) \quad i = 1, \ldots, NUM$

number of pallets in one width of a grid
$AS_i = WG/(WP_i + 2*CLER) \quad i = 1, \ldots, NUM$

number of pallets in one height of a grid
$AH_i = HG/(HP_i + 2*CLER) \quad i = 1, \ldots, NUM$

AL_i, AS_i, and AH_i are rounded off to the nearest lower integer. Where, CLER = half clearance between two pallets.

The total number of pallets of material type i that can be stored in one grid is:

$$CAPG_i = AL_i \times AW_i \times AH_i$$

Total number of grid spaces required for storage of material i is:

$$AP_i = (PP_i/CAPG_i) \quad i = 1, \ldots, NUM$$

AP_i is rounded off to the nearest higher integer.

The number of grid spaces in one rack = RACK = NRxNC

The total number of racks required to store all the material is:

$$RNUM = \sum_{i=1}^{num} AP_i /RACK$$

The total number of blocks formed with RNUM racks is:

BNUM = [RNUM/(2*NRACK)]

BNUM is rounded off to the nearest higher integer.

Estimation of Rack Dimensions

Length of a rack = LR = NR*LG + (NR+1)*CL
Width of a rack = WR = WG + 2*CL
Height of a rack = HR = NC * HG + 2*CL

Step 2: Find the number of blocks along the length of square area by:

P1 = (LS/LB)

Find the number of blocks width-wise in WS by:

Q1 = (WS/WB)

P1 and Q1 are rounded off to the nearest lower integer.

Step 3: Find the total number of blocks that can fit in a square by:

TBLOCK = P1 * Q1

Step 4: If the required number of blocks, BNUM, are more than the number of blocks already arranged, TBLOCK, then the space remaining in the row is calculated. Slack in a row is given by:

SRow = LS - LB * P1

Now the length of the layout is modified so that one more column of blocks can fit in it. The new length of the layout becomes:

LS = LS + (LB - SRow)

Go to Step 2.

If no space is available in the sides of the layout, the new blocks are added as a new row. Hence, the slack in a column is calculated as:

SColumn = WS - WB * Q1

The length of the column is modified so that one more row can fit in it. The new width of the layout becomes:

WS = WS + (WB - SColumn)
Go to Step 2.

If the number of blocks that can be arranged in the layout with modified length and width is less than the required number of blocks, BNUM, then the layout dimensions are estimated as:

Length of Warehouse
LS = P1 * LB
Width of Warehouse
WS = Q1 * WB

Once the blocks have been arranged in near-square fashion, the layout is then surrounded by a highway. Robots can use the highway to access different parts of the layout quickly.

It should be noted that this procedure allows the placing of all the blocks either lengthwise or widthwise. More compact layouts were generated when the blocks were placed both lengthwise and widthwise. These compact layouts were rejected because the combination of blocks created more junctions and more turns for robot movement, which is not desirable.

APPENDIX C:

PATH PLANNING AND COLLISION AVOIDANCE ALGORITHMS FOR A TWO-ROBOT INTERACTING ENVIRONMENT

Path Planning Algorithm

1. Assign the start node and goal node for each robot.

2. Update the tree structure of the global map based on the information on static obstacles. Static obstacles are introduced by removing those branches of the tree that connect the nodes between which the static obstacle is located.

3. Find a path from start node to goal node based on breadth-first search technique on the tree structure.

4. Evoke the collision avoidance subroutine to create a node-like chart and calculate the velocity of movement between each pair of nodes in the paths.

5. Execute movement for each robot until current node is equal to goal node.

6. If no obstacle is traced, move to the next node. Set current node equal to next node. Go to Step 5 of the algorithm.

7. If an obstacle is traced by a robot, update the tree structure by removing branches that connect the current node of that robot with the next node.

8. Conduct a breadth-first search for both the robots from their current nodes to their goal nodes, respectively.

9. Go to Step 4.

General Collision Avoidance Algorithm

1. Starting with the paths chosen for the two robots, establish the node-time chart by calculating the time to reach the nodal points within each path.

2. Compare the node points in the paths.

3. If there are no identical nodes, collision is not possible.

4. Exit the algorithm with the node-time chart unchanged.

5. If there are identical nodes, compare the arrival times at the identical nodes.

6. If the arrival times are different, go to Step 4.

7. If the arrival times are within a tolerance limit specified, there is a chance of collision.

8. Depending on the priority set for the robots, modify the node-time chart of the lower priority robot so that the arrival times at the identical node are increased, thereby making the robot move slower than the allowed speed. Changing the arrival time at a node will automatically alter the arrival times at the subsequent nodes in the path.

9. Go to Step 2.

APPENDIX D:

EXAMPLE OF COLLISION AVOIDANCE ALGORITHM FOR BYPASSING

The various steps of the collision-avoidance algorithm are illustrated in the flow chart given in Figure 8 of the main text.

Consider a three-robot working environment. ROB1 starts from node B3 (refer to Figure 7 of the main text) and moves to the target node B7. ROB2 moves from node B6 to node B10 and ROB3 moves from node C6C8 to node B10. Let ROB2 have the highest priority (=1) and ROB3 has the lowest priority (=3). The path planning algorithm yields the following paths for the robots after carrying out a breadth-first search from their start nodes to the goals.

ROB1: B3 → B3B5 → B5 → B5B7 → B7

ROB2: B6 → B6B8 → B8 → B8B10 → B10

ROB3: C6C8 → C8 → B3B5 → B8 → B8B10 → B10

The general collision avoidance algorithm is then executed to modify the node-time chart for each robot. Table D1 lists the arbitrary node-time chart for this three-robot case.

Note that the assignment of the arrival times to the nodes is based on the assumptions that it takes 20 time units to travel from the intersection nodes (e.g., B3B5) to the main modular nodes (e.g., B5) and 40 time units when traveling from main module nodes (e.g., B5) to intersection nodes (e.g., B5B7). Let ROB1 trace an obstacle while traveling between B3 and B3B5 and decide to bypass. The newly computed path for ROB1 is:

Cur-Pos → B8 → B3B5 → B5 → B5B7 → B7

(Cur-Pos refers to the current position of the robot.)

The search for paths for ROB2 and ROB3 from their current positions to respective goals will yield the following:

ROB2, Cur-Pos → B6B8 → B8 → B8B10 → B10

ROB3, Cur-Pos → C8 → B3B5 → B8 → B8B10 →B10.

The collision avoidance algorithm for bypassing algorithm is now called into operation.

Step 1:

BR = ROB1

N(BR,I) = B8

N(BR,I+1) = B3B5

N(BR,I-1) = Cur-Pos.

SUCC(BR,I) = B8B10

Table D2 shows the node-time chart after executing the general collision avoidance algorithm. Note that the obstacle was traced by ROB1 after traveling for five time units from node B3. The arrival time at Cur-Pos is assigned 35.

Step 2: R = ROB2

Compare the nodes in the path of ROB2 with B8

B8 is a node in the path of ROB2: (R,J) = B8

Priority of ROB2 > Priority of ROB1

Step 3: N(R,J-1) = B6B8; N(R,J-1) ≠ N(BR,I+1)

N(R,J+1) = B8B10 = SUCC(BR,I)

T(R,J) = 60; T(BR,I) = 70; T(R,J) < T(BR,I): Collision is possible

N(R,J +1) = B8B10

T(R,J+1) = 100

N(BR,I-1) = Cur-Pos

Alter T(BR,I-1) to 100 + 5 = 105

Modify node-time chart (Table D3)

Step 2: Consider the path of ROB3

R = ROB3; B8 is a node in the path of ROB3

Priority of ROB3 < Priority of ROB1: N(R,J) - B8

Step 3: N(R,J-1) = B3B5 = N(BR,I+1)

T(R,J-1) = 60; T(BR,I+1) = 160

T(R,J) = 80 T(BR,I) = 140

T(R,J-1) < T(BR,I+1) and

T(R,J) < T(BR,I)

N(R,J+1) = B8B10 = SUCC(BR,I)

T(R,J) (=80) < T(BR,I) (=140)

N(R,J-1) = B3B5

Alter T(R,J-1) to 140 + 5 = 145

Modify node-time chart (Table D4)

T(R,J-1) = 145; T(BR,I+1) = 160

T(R,J) = 165; T(BR,I) = 140

T(R,J-1) < T(BR,I+1) and T(R,J) > T(BR,1)

Alter T(R,J-1) to 160 + 5 = 165

Modify node-time chart (Table D5)

T(R,J) (=185) > T(BR,I) (=140)

No more robots.

Step 4: Execute the general collision avoidance algorithm. Since the arrival times at the identical nodes are different, no alternation of arrival times is necessary. The node-time charts remain the same.
End.

Table D1

Arbitrary Node-Time Chart

ROB1	**Priority = 2**	**ROB2**	**Priority = 3**	**ROB3**	**Priority = 1**
Node	**Arrival Time (in seconds from start)**	**Node**	**Arrival Time (in seconds from start)**	**Node**	**Arrival Time (in seconds (from start)**
B3	30	B6	0	C6C8	20
B3B5	70	B6B8	40	C8	40
B5	90	B8	60	B3B5	60
B5B7	130	B8B10	100	B8	80
B7	150	B10	120	B8B10	120
				B10	140

Table D2

First Adjusted Node-Time Chart

ROB1 Priority = 2		ROB2 Priority = 3		ROB3 Priority = 1	
Node	Arrival Time (in seconds from start)	Node	Arrival Time (in seconds from start)	Node	Arrival Time (in seconds (from start)
Cur-Pos	35	Cur-Pos	35	Cur-Pos	35
B8	70	B6B8	40	C8	40
B3B5	90	B8	60	B3B5	60
B5	110	B8B10	100	B8	80
B5B7	150	B10	120	B8B10	120
B7	170			B10	140

Table D3

Second Adjusted Node-Time Chart

ROB1 Priority = 2		ROB2 Priority = 3		ROB3 Priority = 1	
Node	Arrival Time (in seconds from start)	Node	Arrival Time (in seconds from start)	Node	Arrival Time (in seconds (from start)
Cur-Pos	105	Cur-Pos	35	Cur-Pos	35
B8	140	B6B8	40	C8	40
B3B5	160	B8	60	B3B5	60
B5	180	B8B10	100	B8	80
B5B7	220	B10	120	B8B10	120
B7	240			B10	140

Table D4

Third Adjusted Node-Time Chart

ROB1 Priority = 2		ROB2 Priority = 3		ROB3 Priority = 1	
Node	Arrival Time (in seconds from start)	Node	Arrival Time (in seconds from start)	Node	Arrival Time (in seconds (from start)
Cur-Pos	105	Cur-Pos	35	Cur-Pos	35
B8	140	B6B8	40	C8	40
B3B5	160	B8	60	B3B5	145
B5	180	B8B10	100	B8	165
B5B7	220	B10	120	B8B10	205
B7	240			B10	225

Table D5

Fourth Adjusted Node-Time Chart

ROB1 Priority = 2		ROB2 Priority = 3		ROB3 Priority = 1	
Node	Arrival Time (in seconds from start)	Node	Arrival Time (in seconds from start)	Node	Arrival Time (in seconds (from start)
Cur-Pos	105	Cur-Pos	35	Cur-Pos	35
B8	140	B6B8	40	C8	40
B3B5	160	B8	60	B3B5	165
B5	180	B8B10	100	B8	185
B5B7	220	B10	120	B8B10	225
B7	240			B10	245

APPENDIX E:

USER'S GUIDE FOR SIMULATION PACKAGE

Introduction

Three different simulation packages have been created. System I has two parts. Part I deals with creating the layout using the preprocessor. Part II is for simulation study of the robots in the designed warehouse. The simulation could also be performed on a predefined warehouse that has been created using the preprocessor and stored in the memory.

Simulation System II uses a modular warehouse design that was used for simulation study in the early stages of the research effort. It has all the available options as in Section I for static and dynamic planning.

A preprocessor that was created for the design of warehouses from raw material and pallet dimension data is included under System III.

System I

Part I: Warehouse Layout Creation Using Preprocessor

The following files are required for the operation of the preprocessor.

- BASICA.COM (if using Zenith PC, use Zenith version of BASICA)
- BASICA.EXE
- COMMAND.COM
- WLAYOUT.BAS

1. Load BASICA by typing

   C > <u>BASICA</u>

2. Load preprocessor file once inside BASICA.

   (F3) LOAD "WLAYOUT"

3. Run the preprocessor by pressing function key F2.

The preprocessor takes the user through four steps. Detailed explanations of the steps are provided within the program itself.

The preprocessor creates two files as its output:

- WARE4.PIC is a graphical representation of the created layout
- NODE4.DAT is the generated nodal data base

At the end of the program, exit BASIC by typing "SYSTEM."

Part II: Simulation of the Robots in the Designed Warehouse

Note: Make sure that the two files "WARE4.PIC" and "NODE4.DAT" are available for the simulation. If these files are stored under different names, use the COPY command or RENAME command.

```
C > COPY WARE.EX1 WARE4.PIC

C > COPY NODE.EX1 NODE4.DAT
```

The following files are needed to run the simulation package.

- WSIM.EXE (executable version of the simulation program)
- WSIM.BAS
- COMMAND.COM
- W_SEARC4.COM (search routine for path selection)
- BASRUN20.EXE (Library files)
- BASRUN20.LIB
- WARE4.PIC } (Data files)
- NODE4.DAT }
- CALPAS3.BAT (Batch file for search)
- WARE4.EX1 } (Optional, files for demonstration)
- NODE4.EX1 }

It is important that the key 'Caps Lock' is on so that all user inputs are in capitals. The simulation is menu driven. Help is available by pressing the "H" key.

1. To run the program, type

   C > <u>WSIM</u>

   The system prompts with the Main Menu.

2. Start the simulation by choosing item (1) (use predefined layout).

3. The second step is to choose static planning (item 2).

In static planning, you may allocate tasks to robots, introduce/remove static constraints and display/select paths.

Note that for every node selected, you need to specify the block number and the node number inside each block. For each block, the nodes are numbered 1, 2, 3, and 4 in a clockwise direction. For example, in Figure E1, the letter B represents Block No. 1, Node No. 2. After path selection, exit the static planning by choosing Main Menu (item 5).

4. Choose dynamic planning (item 3) from main menu for dynamic simulations. The various options inside dynamic simulation are provided in the "Help" menu.

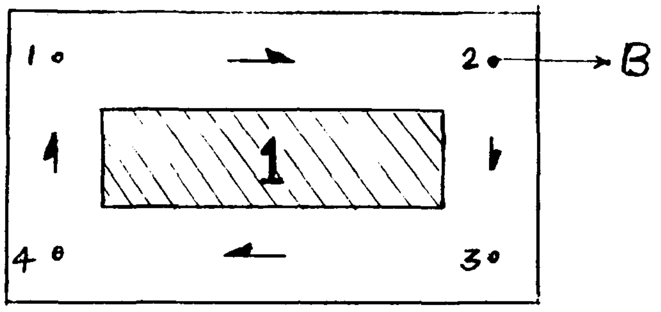


Figure E1. Node numbering.

5. Once the robots have reached their respective targets, press the "return" key to get back into Main Menu.

6. Always "End Simulation" (item 4) before starting a new simulation.

Demonstration data files have been created under "WARE4.EX1" and "NODE4.EX1". To use this for demonstration, execute the following statements:

```
C > COPY WARE4.EX1  WARE4.PIC
C > COPY NODE4.EX1  NODE4.DAT.
```

Note: Files previously stored under WARE4.PIC and NODE4.DAT will be erased and hence have to be stored under a different name before execution of these commands.

System II

System II allows the simulation study to be carried out in a predefined modular warehouse. This system allows both graphical simulation and status monitoring to be carried out on the same screen.

The following files are required for this simulation:

- COMMAND.COM
- BASRUN20.LIB (Library File)
- BASRUN20.EXE
- WA1.BAS } (Simulation file)
- WA1.EXE
- CALPAS.BAT (Batch file for search)
- W.SEARCH.COM. (search routine)
- TTREE1 (node data file)

1. Start simulation by typing

   C > WA1.

The simulation is menu-driven. Select "Predefined Layout" (item 1) to create the layout picture. The static and dynamic planning follow the same steps as explained in Part II of System I.

2. Always "End Simulation" (item 4) before starting a new simulation.

Note: The modules are numbered as A1, A2, B1, C1, etc. The junctions on the highway are labeled as A, B, AA, BB, AX, etc. The mainway nodes are identified as shown in Figure E2. The color simulator shows traffic moving from left to right and from bottom as pink lines. Movement from left to right and from bottom to top is shown by white lines.

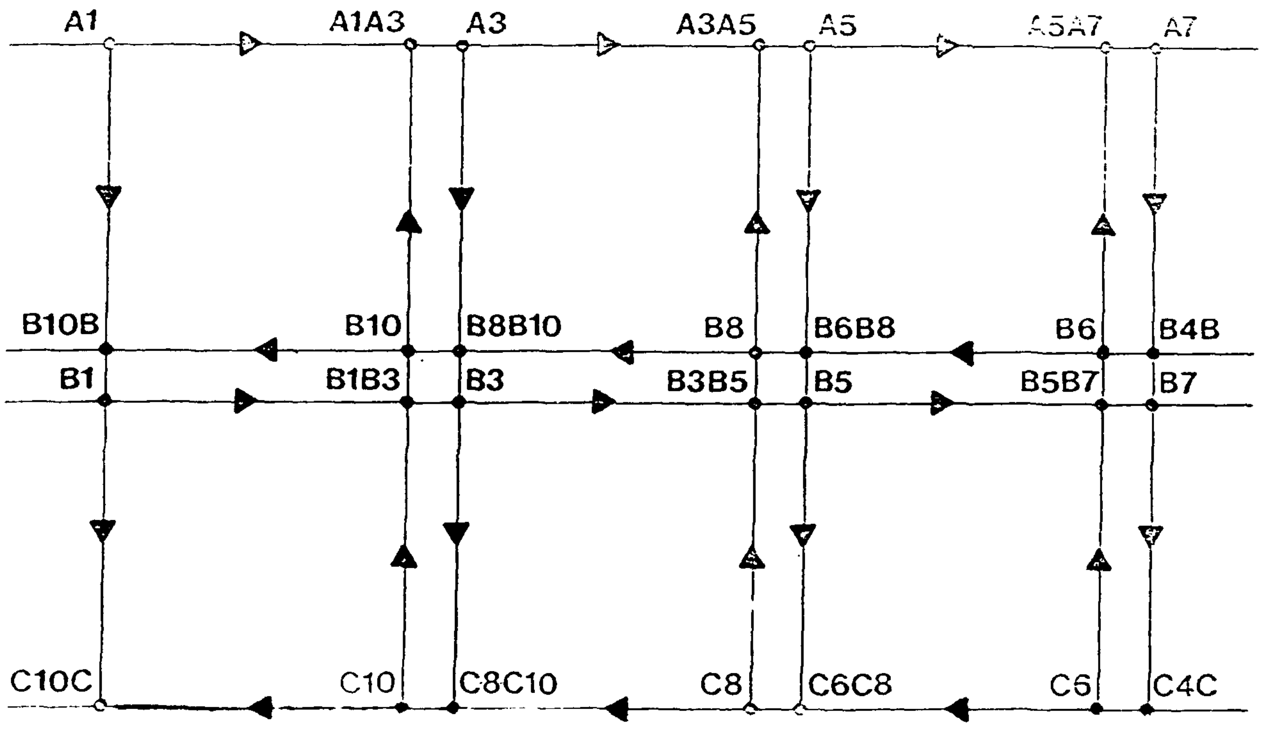


Figure E2. Mainway nodes and direction of traffic.

System III

A preprocessor for the design of warehouse layouts is provided by this system. The user inputs the various dimensions required to design the warehouse. The user has two options for inputting data. Option 1 makes use of an input file. (A sample input file is provided under "WARE.DAT".) In Option 2, the user provides data using keyboard inputs by answering questions.

The following files are required:

- COMMAND.COM
- BASRUN20.EXE
- BASRUN20.LIB (Library file)
- WA2.EXE } (Simulation files)
- WA2.BAS }
- W.SEARC2.COM (Search routine)
- WARE.DAT (Sample data file)
- CALPAS1.BAT (Batch file for search)

1. Start the preprocessor by typing

   C > <u>WA2</u>

2. Choose the 'create layout' (item 1) option to design the layout.

3. The various steps are menu-driven and follow the same pattern as in System I and II.

USACERL DISTRIBUTION

Chief of Engineers
ATTN: CEHEC-IM-LH (2)
ATTN: CEHEC-IM-LP (2)
ATTN: CERD-L

CEHSC
ATTN: Library 22060

US Army Engineer Districts
ATTN: Library (40)

US Army Engr Divisions
ATTN: Library (14)

US Military Academy 10996
ATTN: Facilities Engineer
ATTN: Dept of Geography & Computer Sciences
ATTN: MAEN-A

CECRL, ATTN: Library 03755

CEWES, ATTN: Library 39180

NAVFAC
ATTN: Naval Civil Engr Lab, Library Code L08A

Engineering Societies Library
New York, NY 10017

US Government Printing Office 20401
Receiving/Depository Section (2)

Defense Technical Info. Center 22304
ATTN: DTIC-FAB (2)