Blanche—An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle

Ingemar J. Cox, *Member, IEEE*

Abstract—**This paper describes the principal components and capabilities of Blanche, an autonomous robot vehicle. Blanche is designed for use in structured office or factory environments rather than unstructured natural environments. This is a significant restriction, but still allows for many potential applications. An overview of the physical configuration of the vehicle is presented as well as a description of its two sensors, an optical rangefinder and odometry. It is assumed that an off-line path planner provides the vehicle with a series of collision-free maneuvers, consisting of line and arc segments, to move the vehicle from a current to a desired position. On board the vehicle, the line and arc segments specifications are sent to control software consisting of low-level trajectory generation and closed-loop motion control. The trajectory generator takes each segment specification and generates a reference vector at each control update cycle. The cart controller controls the front steering angle and drive velocity using conventional feedback compensation to maintain small errors between the reference and measured states.**

The controller assumes accurate knowledge of the vehicle's position. We believe that position estimation is a primary problem that must be solved for autonomous vehicles working in structured environments. Blanche's position estimation system consists of 1) an *a priori* map of its environment, represented as a collection of discrete line segments in the plane; 2) a matching algorithm that registers the range data with the map (of line segments) (this algorithm has the properties that it is robust against missing and spurious data and is reasonably fast allowing matching to occur very frequently (approximately every 8 s), and 3) the matching algorithm also estimates the precision of the corresponding match/correction that is then optimally (in a maximum likelihood sense) combined with the current odometric position to provide an improved estimate of the vehicle's position. The vehicle and associated algorithms have all been implemented and tested within a structured office environment.

Except for the off-line global path planner, the entire autonomous vehicle is self-contained, all processing being performed on board and with no recourse to passive or active beacons placed in the environment. We believe this vehicle is significant not just because of the sensing and algorithms to be described, but also because its implementation represents a high level of performance at low cost.

I. Introduction

BLANCHE [4] is an experimental vehicle designed to operate autonomously within a structured office or factory environment. We believe that a key problem for such a vehicle is accurate position estimation, particularly if active or passive beacons are not permitted. Blanche is an experiment in the guidance and navigation of an autonomous vehicle within an office environment. From a research perspective, Blanche has served as a testbed with which to experiment with such things as real-time programming languages, sensor integration/data fusion techniques, and with techniques for error detection and recovery. From a commercial perspective, we have tried to be cost conscious, since many potential commercial applications are very cost sensitive.

This paper describes the principal components and capabilities of the vehicle. Section II describes the physical structure of the vehicle and its associated sensors. Section III describes the trajectory generation and guidance control system for the vehicle. Experimental results are included. The guidance system compares the reference position and velocity vector with the measured vector and uses conventional linear feedback for control. Section IV describes the position estimation system employed by the vehicle. Included in this is a description of the robot's map representation, Section IV-A, the algorithm used to match the range data to this map, Section IV-C, as well as some implementation details. Section IV-D discusses how the odometry position estimate is combined with the correction estimated by the matcher. The system ahs been implemented on the vehicle and experimental results are included in Section IV-E. Section V contains a brief description of the software implementation.

Manuscript received September 4, 1989; revised September 1, 1990. A portion of this work was presented at the IEEE International Workshop on Intelligent Robots and Systems, 1989.

The author is with the NEC Research Institute, Princeton, NJ 08540.

IEEE Log Number 9041908.

II. Overview of Blanche

Blanche, shown in Fig. 1, has a tricycle configuration consisting of a single steerable drive wheel at the front and two passive rear wheels. The vehicle is powered by two sealed 12V 55Ah batteries which, in the current configuration, provide a useful lifetime of approximately 7 h. Control of the cart is based on a Multibus system consisting of a MC68020 microprocessor with MC68881 match coprocessor, 2 Mbyte of memory, an ethernet controller, a custom two-axis motor controller, and an analog-to-digital convertor. The Motorola 68020 runs a real-time UNIX® derived executive called NRTX [16].

Blanche is designed to be low cost, and its dependence on only two sensors, an optical rangefinder and odometry, reflects this. Both sensors are low cost and together provide all the navigational sensing information available to the cart. The advantage of odometry is, of course, that it is both simple and inexpensive. However, it is prone to several sources of errors. First, surface roughness and undulations may cause the distance to be over estimated. Second, wheel slippage can cause distance to be under estimated. Finally, variations in load can distort the odometer wheels and introduce additional errors. If the load can be measured, then the distortion can be modeled and corrected for [28]. Where appropriate, a simple and more accurate alternative is to provide a pair of knife-edge nonload-bearing wheels solely for odometry. Blanche uses this approach. However, even very small errors in the vehicle's initial position can lead to gross errors over a long enough path. Consequently, it is imperative the vehicle's environment be sensed in order to anull these accumulating errors.

1042-296X/91/0400-0193$01.00

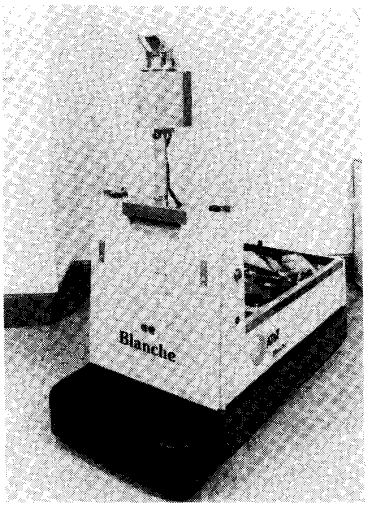


Fig. 1. Cart Blanche.

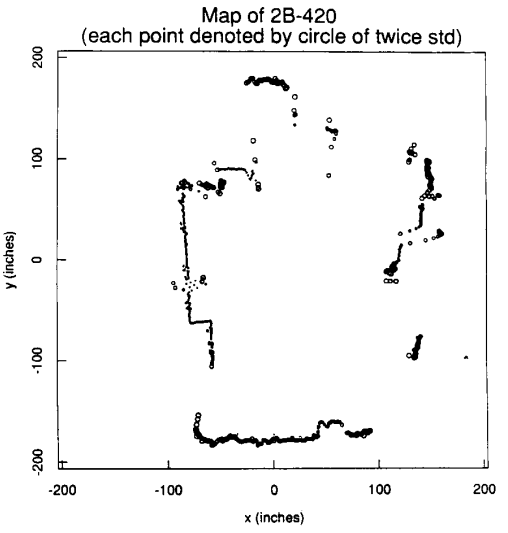


Fig. 2. Range map. The radius of each circle equals twice the standard deviation at each point.

A simple low-cost optical rangefinder has been developed specifically for cart navigation [20]. The rangefinder uses an approximately 1-in-diameter beam and a rotating mirror to provide 360° polar coordinate coverage of both distance and reflectance out to about 20 ft. Both radial and range resolution correspond to about 1 in at a ranging distance of 5 ft, with an overall bandwidth of approximately 1 kHz. Thus, there is the capability to extract approximately 1000 samples per revolution, though, in practice, we limit the number to about 180.

Fig. 2 shows a typical range map of a room obtained from a single scan of the rangefinder. A scan typically takes about 1 s. Each point is represented by its corresponding region of uncertainty, denoted by a circle of radius twice the standard deviation in the measurement. It should be pointed out that the error, due to assuming that the range output is linear with distance, may sometimes exceed the error due to noise in the rangefinder. This systematic error can be removed by using a table look up technique to accurately map range output into distance.

We assume that an off-line path planner [31] generates a series of collision free maneuvers, consisting of line and arc segments, to move the vehicle from a current to a desired position. Since the cart has a minimum turning radius (approximately 2 ft), it is not possible to simply turn on the spot and vector to the desired position as is the case for differentially driven vehicles. This path is downloaded to the vehicle, which then navigates along the commanded route. The next section describes how these line and arc segments are interpreted by the vehicle.

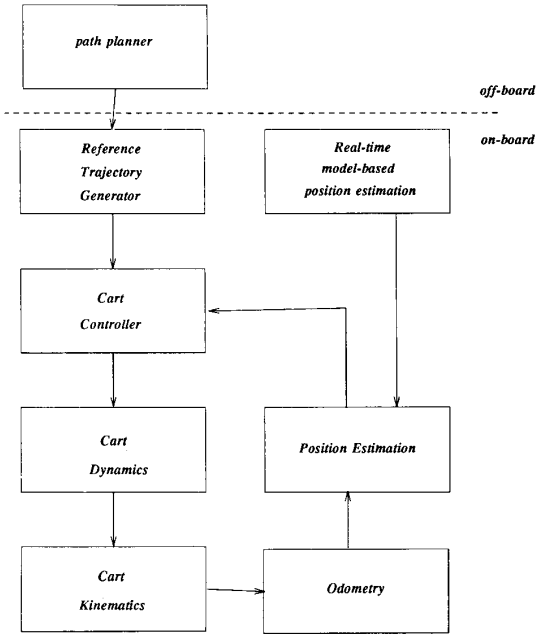


Fig. 3. Block diagram of the low-level control system.

III. Guidance (Trajectory Generation and Control)

This section is taken from Nelson and Cox [23]. The path plan received from the off-board path planner consists of a list, each line of which specifies a segment of the path. The segment specification includes the segment type and the desired state at the end of the segment. The segment specifications are sent to control software consisting of low-level trajectory generation and closed-loop motion control, as illustrated in Fig. 3 [23]. Briefly, the trajectory generator takes each segment specification and generates a reference vector at each control update cycle (every 0.1 s). The cart controller controls the front steering angle and drive velocity using conventional feedback compensation to maintain small errors between the reference and measured states, as described in Section III-A.

The trajectory generator (TG) continually provides an appropriate reference state for the cart controller to track. The TG computes at each control update cycle the reference state on the path, the reference steering angle and drive speed, and the distance to the end of the current path segment. The reader is directed to [23] for a detailed explanation of the generation of the reference vectors. When the distance to the end-point of the segment becomes less than the distance between the preceding reference points, the TG requests the next segment in the path

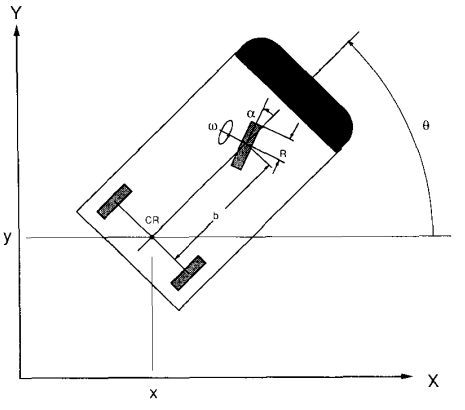


Fig. 4. Plan view of the cart coordinates.

plan and begins generating reference states along the new segment—lines or continuous curvature splines [22].

The main advantage of the reference path guidance scheme over point-to-point guidance schemes [13], [29] is that the cart's on-board system is always providing a current (computed) point on the path where the cart should be, with smooth transitions across path-segment boundaries. When the distance between this computed current position and the end of the path segment is less than a threshold, a transition from current to next path segment occurs. A point-to-point control scheme, compares the current *measured* position (rather than current computed position) to the desired position. However, because of measurement errors, such a scheme may not detect the point at which the control algorithm should switch to a new target point, and hence may fail to make the proper transitions from one path segment to the next.

The TG scheme also provides a means for onboard monitoring of the operating status of the cart. Under normal operating conditions, the difference between the measured state of the cart and the current reference state should be small. If this state error ever exceeds some established limits, then the cart must be functioning abnormally. The cart controller can test for abnormal errors and take remedial action. By this means, system malfunctions, such as failure of motors or odometry sensors, can be detected in time for the cart to be safely stopped.

A third advantage of the TG scheme is that it provides a convenient separation between the path-guidance logic and the feedback control logic, which must be tuned to the characteristics of the closed-loop dynamics of the steering and drive systems of the cart. Once properly adjusted, the controller design remains fixed throughout all the different operating situations that may arise along the path; all the controller needs to know is what the current reference and measured states are. The TG takes care of dealing with the geometric computations necessary to evolve the path plan as points in time, and also with anomalies that may require local path modifications, either spatially, temporally, or both.

A. Cart Controller

The coordinates by which the cart is controlled are the front-wheel steering angle α and the front-wheel drive speed ω. If the location of the cart is measured with respect to the center of rotation (CR) of the cart, located at the midpoint between the rear odometry wheels (see Fig. 4), the speed v, heading θ, and position (x, y) of the cart are related to the steering angle and drive speed coordinates (α, ω) by the kinematic equations

$$\begin{aligned} v &= R\omega \cos \alpha \\ \theta &= (R/b)\omega \sin \alpha \\ \dot{x} &= v \cos \theta \\ \dot{y} &= v \sin \theta \end{aligned} \tag{1}$$

where R is the radius of the drive wheel and b is the wheelbase, as shown in Fig. 4.[1]

The task for the cart controller is to steer and drive the front wheel of the cart so that the CR point accurately tracks the reference points specified by the trajectory generator. The cart position and velocity measurements necessary for performing this task are obtained from the position estimation system indicated in the cart control block diagram shown in Fig. 3.

The cart controller consists of a path controller and two motor-control units. The motor-control units provide inner control loops for the steering and drive motors, while the path controller provides the outer control loop for trajectory tracking.

1) Motor Error Control: The front-wheel steering angle and the front-wheel drive speed depend on the motor-load dynamics in response to the control signals to the motors. The purpose of the motor-control units is to compensate for the dynamics in such a way that the motor outputs (α, ω) respond rapidly and smoothly to accurately follow the *commanded* values (α_c, ω_c) output from the path controller. This is done by conventional feedback compensation, based on error signals between the commanded values and the motor outputs values, measured by the angle encoders on the motor shafts. To achieve reasonably high-gain error control without causing undesirable, or even unstable response, controller chips containing digital pole-zero compensating filters are used.

The dynamics of the motor-load combination for the two units are represented to a close approximation by the linear model

$$\begin{aligned} \ddot{\alpha} &= a_1 - H_1\dot{\alpha} \\ \dot{\omega} &= a_2 - H_2\omega \end{aligned}$$

where a_1 and a_2 are the control accelerations (torque per unit moment of inertia) developed by the steering and drive motors, respectively, and H_1 and H_2 are the friction coefficients of the two motor-load systems. The digital error signals driving the control units are

$$e_\alpha(k) = C_r(\alpha_c(k) - \alpha(k))$$

and

$$e_\omega(k) = C_r(\omega_c(k) - \omega(k))$$

where k denotes the kth sampling period, and C_r is the angle encoder gain (in counts per radian). The difference between the two control units is that one is controlling the angle while the other is controlling the angular velocity. Because of the extra integration in the angle control dynamics, the steering angle is the more difficult one for which to compensate properly.

2) Path Error Control: The path controller generates the steering and drive command signals (α_c, ω_c), using the reference values (α_r, ω_r) from the TG, and path-errors derived from the measured state of the cart relative to the reference state, as

[1] The choice of x, y position determines what point on the cart tracks the reference path, as well as the form of the kinematic equations [1]. The choice to have the CR point track the path, rather than, say, the front wheel, was made for reasons related to path planning and cart maneuverability [21].

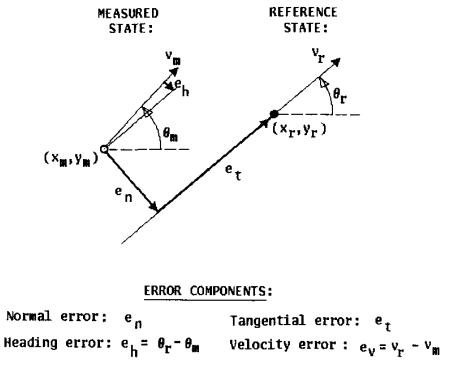


Fig. 5. Error components used for path control.

shown in Fig. 5. The path error is resolved into four components. As diagramed in Fig. 5, the distance error is resolved into *tangential* error e_t and *normal* error e_n

$$e_t = (x_r - x_m)\cos\theta_r + (y_r - y_m)\sin\theta_r$$
$$e_n = -(x_r - x_m)\sin\theta_r + (y_r - y_m)\cos\theta_r.$$

The velocity error is resolved into *heading* error e_h and *speed* error e_v

$$e_h = \theta_r - \theta_m$$
$$e_v = v_r - v_m.$$

Given the reference values α_r, ω_r and these error components, the command steering and drive signals generated by the path controller in each update cycle are

$$\alpha_c = \alpha_r + C_1 e_n + C_2 e_h$$
$$\omega_c = \omega_r + C_3 e_t + C_4 e_v. \quad (2)$$

Note that when the cart is following the reference path without error, the command steering and speed are equal to their reference values.

The weighting constants $C_1, \cdots, C_4$ in (2) are chosen to optimize the overall path-tracking performance of the cart. As with the motor control units, it was the steering control part of the outer loop design that was the most sensitive to the choice of the control weights in (2). Because the cart speed affects the optimum steering gains through the kinematic relations of (1), work is currently under way to find an appropriate functional dependence on speed for these weights, such that path control is maintained at near-optimum during the critical docking segments, when the speed is being reduced to zero. Although not included in the experimental results described below, a quick analysis suggested that the gains C_1 and C_2 should be scaled by the inverse of v^2 and v, respectively, where v is the speed of the vehicle. This has been implemented and appears to provided reasonable results over the range of tested speeds from 3 to 12 in/s.

B. Experimental Results

The test runs for the vehicle were limited to a 14 × 12 ft "workspace" in the robot laboratory. Hence, the test paths have been mostly figure-eight patterns with tight turns in which the steering angle is at or near the hard limit of 45°. Fig. 6 illustrates the performance of the cart over such a test path. The reference trajectory is the solid line, while the cart's measured trajectories are shown for three operating speeds: 2 in/s (dotted), 4 in/s (dashed), and 6 in/s (large dashed). In these experiments, "measured" refers only to the vehicle's odometric estimate of the vehicle's position. The control algorithm clearly performs better at the slower speeds as the gains have been optimized for the slower speeds. The performance degradation at higher speeds is usually acceptable since high-speed transit maneuvers typically do not require as much precision as low-speed docking maneuvers. Some overshoot at the beginning of the arc segments is unavoidable, since the control cannot make the steering wheel follow the angle discontinuity at the transition point between line and arc segments. Furthermore, as the vehicle enters the tighter loop on the right, the overshoot at high speeds is exaggerated due to the steering wheel hitting the hard limit step.

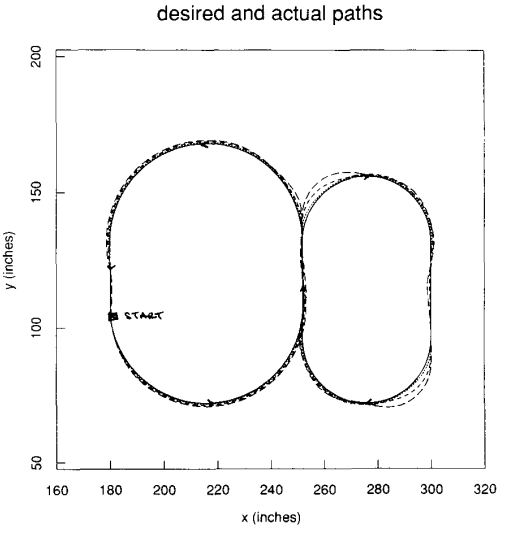


Fig. 6. Measured path trajectories for the vehicle.

The discontinuity in steering angle could be avoided if we were to control the point directly under the drive wheel rather than the midpoint between the rear wheels. However, controlling the orientation of the vehicle becomes more complex, as does path planning. A further alternative would be to replace the arc segments with segments of curves that are continuous in curvature. We have recently incorporated "polar splines" [21] into the trajectory generator, which closely approximate the optimum minimum square jerk curves of Kanayama and Hartman [15]. In fact, the trajectory generator is easily extensible, as discussed in Section VI.

IV. Navigation (Position Estimation)

The guidance control subsystem assumes an accurate knowledge of the vehicle's position. By position, we mean the vehicle's (x, y, θ) configuration with respect to either a global or local coordinate frame, *not* topological position, e.g., to the left of the wall. Dead reckoning using odometry sensors drifts with time so that the estimated position of the vehicle becomes increasingly poor. This complicates the process of position estimation. In order to correct for these cumulative errors, the vehicle must sense its environment and at least recognize key landmarks. Using sensory information to locate the robot in its environment is the most fundamental problem to providing a mobile robot with autonomous capabilities. One of the goals of this research has been to develop robust algorithms that will

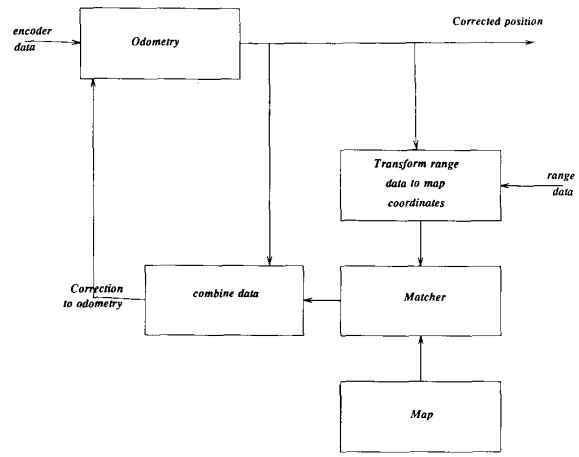


Fig. 7. Expanded view of position estimation subsystem.

allow a vehicle to navigate in a structured office or factory environment.

Navigation can be broadly separated into two distinct phases, reference and dead reckoning, as discussed in Cox and Wilfong [9]. Reference guidance refers to navigation with respect to a coordinate frame based on visible external landmarks. Dead reckoning refers to navigation with respect to a coordinate frame that is an integral part of the guidance equipment. Dead reckoning has the advantage that it is totally self-contained. Consequently, it is always capable of providing the vehicle with an estimate of its position. Its disadvantage is that the position error grows without bound unless an independent reference is used to periodically reduce the error. Reference guidance has the advantage that the position errors are bounded, but detection of external references or landmarks and real-time position fixing may not always be possible. Clearly dead reckoning and external reference navigation are complementary and combinations of the two approaches can provide very accurate positioning systems.

Position estimation based on the *simultaneous* measurement of the range or bearing to three or more known landmarks is well understood [26]. However, recognizing naturally occurring reference points within a robot's environment is not always easy due to noise and/or difficulties in interpreting the sensory information. Placing easy to recognize beacons in the robots workspace is one way to alleviate this problem. Many different types of beacons have been investigated including 1) corner cubes and laser scanning system, 2) bar-code, spot mark, or infrared diodes [27] and associated vision recognition systems [12], and 3) sonic or laser beacon systems. We chose not to rely on beacons, believing that the ability to operate in an unmodified environment was preferable from a user standpoint.

There have been many efforts to use high-level vision, particularly stereo vision [2], [18] by which to navigate. However, conventional vision systems were ruled out because of the large computational and associated hardware costs: We want the vehicle to be economic. Fig. 7 is an overview of Blanche's position estimation system. It consists of:

1) an *a priori* map of its environment,
2) a combination of odometry and optical range sensing to sense its environment,
3) an algorithm for matching the sensory data to the map [7], and
4) an algorithm to estimate the precision of the corresponding match/correction [8] that allows the correction to be optimally (in a maximum likelihood sense) combined with the current odometric position to provide an improved estimate of the vehicle's position.

Provided the error models are accurate, the combined position estimate is less noisy than any one of the sets of individual measurements. The sensor integration process can, of course, be routinely mechanized by use of a Kalman filter [11], [19], but is not explicitly used in this system.

A. Map Representation

Many spatial representations have been proposed. However, it is worth reflecting on the purpose of a map representation. Our purpose is to compare sensed range data to a map in order to refine our position estimate. The map is *not* intended to be used for path planning; it is not even necessarily intended to be updated by sensory data. Its sole purpose is for position estimation in an absolute coordinate frame. While many spatial representations have been proposed, few appear to have been tested on a real vehicle. One major exception is occupancy grids [10]. Occupancy grids represent space as a 2- or 3D array of cells, each cell holding an estimate of the confidence that it is occupied. A major reason given for not using a more geometric representation is that sensor data is very noisy, making geometric interpretation difficult. This is especially true of sonar data (which Elfes [10] and others used) and is one reason why sonar was not used on Blanche.

The infrared range data is much less noisy. This combined with the fact that factory or office buildings are easily described by collections of line segments also influenced our choice of representation. We represent the environment as a collection of discrete line segments in the plane. A 2D representation was chosen because 1) much of the robot's environment is uniform in the vertical direction (there is not much to be gained from a 3D representation), 2) the range sensor currently provides only 2D information (r, θ), and 3) matching sensor data to 2D maps is significantly simpler than matching to 3D maps. More pragmatically, a line segment description was chosen because a matching algorithm had been developed that used line segments. Moreover, the matching algorithm use (see Section III-C) does not require the explicit extraction of any features from the image.

B. Sensor Data

Sensing of the environment is quite straightforward, relying on odometry and an infrared rangefinder, as previously mentioned. The choice of a rangefinder was based primarily on the belief that ultrasonic rangefinders were poor, with severe problems due to specular reflection. A characteristic of the position estimation system described here is that the vehicle does not move under odometric control for a period, then stop to locate beacons, update is position, and continue. Rather, its position is constantly being updated as the vehicle moves. Since range data are collected while the vehicle is moving, there is a need to remove any motion distortion that may arise. This is done by reading the odometric position at each range sample. The current position and range value are then used to convert the data point into world coordinates for matching to the map.

If (r, α) is the range and angle pair from the rangefinder's local coordinate frame, its position in world coordinates is given

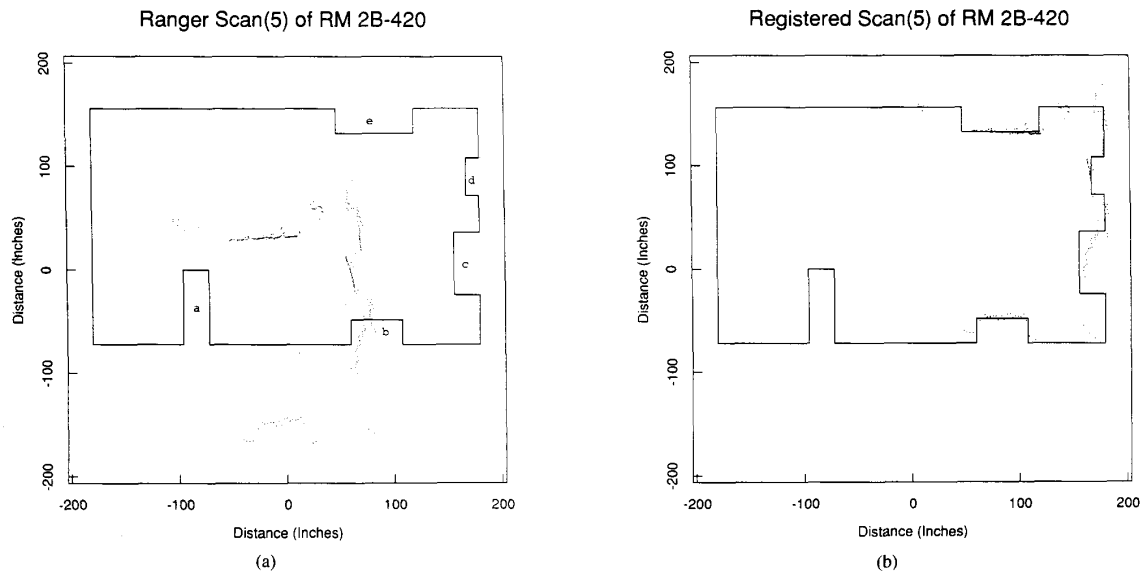


Fig. 8. Map and range data (a) before registration and (b) after registration.

by

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \boldsymbol{CR} \begin{pmatrix} r\cos(\alpha) \\ r\sin(\alpha) \\ 1 \end{pmatrix}$$

where $\boldsymbol{R}$ is the homogeneous transformation describing the relative position of the rangefinder with respect to the cart, and $\boldsymbol{C}$ describes the vehicle's position with respect to the base coordinate system.

C. Matcher

We now describe how the matching of range data to the map is achieved. First, an example. The solid line in Fig. 8(a) is the line segment description of a laboratory room in which Blanche is moved. The model, consisting of 24 line segments, was constructed very simply from measurements based on the 1-ft^2 floor tile grid. Item *a* is a tall desk, item *b* is a ventilation duct, item *c* is some cardboard posters, item *d* is a small refrigerator, and item *e* is a large cupboard. It should be pointed out that the model is very simple; some items in the room are not modeled, and others are only roughly approximated, such as item *c*, which in fact is made up of several cardboard sections all at different distances from the wall. The dotted points show the image based on range data acquired from a single scan of the sensor. In this example there is a small degree of rotation and a large translation, of the order of 9 ft in x and 8 ft in y. Fig. 8(b) shows the results of applying the algorithm. It is evident that the correct congruence has been found.

The general matching problem has been extensively studied by the computer-vision community. Most work has centered upon the general problem of matching an image of arbitrary position and orientation relative to a model. Matching is achieved by first extracting features followed by determination of the correct correspondence between image and model features, usually by some form of constrained search. Once the correspondence is known, determination of the congruence is straightforward.

The following observations motivated the derivation of the matching algorithm described below:

1) The displacement of the image relative to the model is small, i.e., we roughly know where we are at all times. This assumption is almost always true for practical autonomous vehicle applications, particularly if position updating occurs frequently.
2) Feature extraction can be difficult and noisy. Ideally, we would like a matcher that does not require a feature extraction preprocessing stage but works directly on the range data points.

This section is taken from [7] and [8]. If the displacement between image and model is small, then for each point in the image its corresponding line segment in the model is very likely to be its closest line segment in the model. Determination of the (partial) correspondence between image points and model lines reduces to a simple search to find the closest line to each point. The central feature of the original conception was devising a method to use the approximately correct correspondence between image *points* and model *lines* to find a congruence that greatly reduces the displacement. Iterating this method leads to the following algorithm (for actual use, the algorithm incorporates certain computational simplifications as described below):

1) For each point in the image, find the line segment in the model that is nearest to the point. Call this the *target*.
2) Find the congruence that minimizes the total squared distance between the image points and their target lines.
3) Move the points by the congruence found in (2).
4) Repeat steps 1–3 until the procedure converges. The composite of all the step 3 congruences is the desired total congruence.

The rationale for our method of performing step 2 is explained with the aid of Fig. 9. The model shown consists of two line segments. The image consists of several points from both segments that have been displaced by translation up and to the

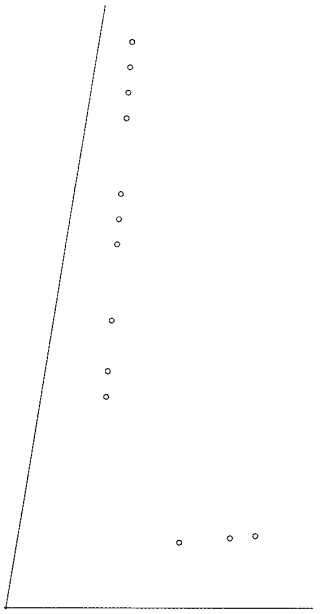


Fig. 9. Example of line model and image points.

right and by a slight rotation counterclockwise. For every point in the image, its target segment is the correct segment, as one would hope. It is natural to seek the congruence that minimizes the total squared distance from the model points to their target segments, i.e., the minimizing procedure tries to move the image points from each segment so that they lie on that segment, but it does not care where on the segment they go. In this case it is possible to reduce the total squared distance to 0, and there is a unique congruence that does so, namely, the inverse of the displacement used in forming the image, and one application of steps 1–3 perfectly recovers the desired congruence. If the original displacement was larger and/or the image contained points from near the ends of the line segments, the correspondence of image points to line segments might be imperfect and several iterations might be required. If the image points contain some error, a potentially infinite iteration process might be required, though in practice only a few iterations usually achieve convergence to sufficient accuracy.

Step 2 is computationally the most complex step. For computational efficiency, two modifications are introduced and the new version is called step 2′. First, each target is changed from a line segment to the infinite line containing the segment. Note, however, that step 1 continues to use the finite segments. Second, the dependence of the moved points on θ is nonlinear, so this dependence is approximated by the first-order terms in θ. Such an approximate congruence is called a pseudocongruence. Note, however, that step 3 continues to use a real congruence, namely, the congruence with the same (t_x, t_y, θ) as the pseudocongruence found in step 2′. Since the algorithm is iterative and the final iterations involve vanishingly small displacements, the approximations involved in these two modifications do not cause error in the final result.

Any congruence can be described as a rotation by some angle θ followed by a translation by t and can thus be denoted by (t, θ). Let the ith point of the image be $v_i = (v_{i1}, v_{i2})'$, where the prime indicates transposition, so that v_i is a column vector, not a row vector. Let $\bar{v}$ be the center of gravity of the image points. Let u_i be a unit vector that is orthogonal to the (infinite line) target of point v_i. Let r_i be the dot product of any point in the target with u_i, so the target consists of all points z such that $u_i \cdot z = r_i$.

Counterclockwise rotation by θ around $\bar{v}$ means moving a point z to

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}(z - \bar{v}) + \bar{v}.$$

The first-order approximation, which is called a pseudorotation, means moving z to

$$\begin{bmatrix} 1 & -\theta \\ \theta & 1 \end{bmatrix}(z - \bar{v}) + \bar{v} = \theta\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}(z - \bar{v}) + z.$$

Then the pseudocongruence used in step 2′ moves point v_i to

$$t + \theta\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}(v_i - \bar{v}) + v_i.$$

The distance of this point from the target line is the dot product of this vector with u_i, minus r_i. Thus, by making the following definitions

$$\begin{aligned} x_{i1} &= u_{i1} \\ x_{i2} &= u_{i2} \\ x_{i3} &= u_i \cdot \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}(v_i - \bar{v}) \\ y_i &= r_i - u_i \cdot v_i \\ b &= (t_x, t_y, \theta) \end{aligned}$$

step 2′ can be described as minimizing the sum of squared deviations from

$$(x_{i1}, x_{i2}, x_{i3})b \simeq y_i.$$

Now suppose the image contains n points. Let X be the $n \times 3$ matrix whose rows are (x_{i1}, x_{i2}, x_{i3}), and let y be the column vector with n components y_i. Then the desired estimate $(\hat{t}_x, \hat{t}_y, \hat{r}_\theta)$ is the least-squares solution to

$$Xb \simeq y.$$

By definition, this constitutes least-squares linear regression. The least-squares solution $\hat{b}$ to this equation can be obtained by solving the equation

$$(X'X)\hat{b} = X'y. \tag{3}$$

The implementation details are given in the following. A major criticism of least-squares linear regression is its sensitivity to outliers. The algorithm as described so far is intrinsically robust against incompleteness of the image, i.e., missing data points. To make it robust against spurious data (outliers), e.g., people walking by, and incompleteness of the model, it is modified further by deleting from consideration in step 2′ points whose distance to their target segments exceed some limit. Note that, for the general matching problem, in which the assumption of small displacement is not necessarily valid, such an approach would not be possible. Our experience is that robustness if very desirable, yet very hard to achieve. The robustness of this algorithm is one of its major features.[2]

[2] Minimizing the median absolute deviation would probably lead to a more inherently robust algorithm [14]. However, it is computationally more expensive.

Step 1 of the algorithm requires that, for each point, its corresponding (closest) line segment be found. Since the map is currently unordered, this entails exhaustively calculating the distance of *every* line from each point in order to find the closest line segment. If there are p points and l line segments, the complexity of this operation is $O(pl)$. However, if the map is first preprocessed to compute the corresponding Voronoi diagram (a step that takes $O(l \log(l))$ time) determination of the closest line segment can then be achieved in $O(p \log(l))$ [24]. Presently, we have not implemented the Voronoi solution.

Since determination of the partial correspondence can quickly dominate the processing time, it pays to keep p and l small. We therefore restrict the number of image points to approximately 180 out of a possible 1000. Originally the map was small enough that we did not have to worry about the size of l, the number of line segments. However, this is no longer the case. Fortunately, our knowledge of the vehicle's position allows us to extract only those lines within say a 20-ft radius. It is this subset of the entire map, which often only numbers four or five, that is then used by the matching algorithm.

Currently, position updates occur approximately every 8 s for an image size of 180 points and a map of 24 line segments. This is sufficient for our requirements. However, it should be noted that the matching is the *lowest* level priority process running on the uniprocessor system. A considerable improvement in speed would be gained by simply dedicating a processor board to this task.

Finally, we note that the matcher returns an (x, y, θ) correction based on an origin at the centroid of the image/range scan. A coordinate transformation is therefore required to convert the matcher's centroid centered correction to a correction based on the vehicle's current position.

D. Integrating Odometric and Matched Positions

It is assumed that the x, y, and θ position and orientation are independent of one another. We therefore describe how the x position is updated, y and θ being identical. We first need to estimate the standard deviation in the measurement of x for both the matcher and odometry.

1) Estimating Matcher Accuracy: Although the rangefinder has an associated measurement uncertainty in (r, α), we do *not* directly transform this uncertainty into y_i. Rather, the measurements are simply transformed into (x, y) coordinates with an assumed error having mean 0 and unknown variance. Because the matching algorithm constitutes a least-squares linear regression, then, referring to (3), if X is known and each y_i is observed with an error having a mean of 0 and unknown variance σ^2, and if the errors are independent variables having the Gaussian (i.e., normal) distribution, then the least-squares solution of this equation has the Gaussian distribution whose mean value is the true underlying value and whose estimated covariance matrix is given by $s^2(X'X)^{-1}$, where s^2 is the unbiased estimate of σ^2,

$$s^2 = \left(y - X\hat{b}\right) \cdot \left(y - X\hat{b}\right)/(n-4).$$

The covariance matrix is calculated as part of the overall calculation for the congruence [8].

2) Estimating Odometry Accuracy: Estimation of the standard deviation for odometry is less rigorous. Presently, it is assumed to be directly proportional to the distance moved. Empirically, we have set the constant of proportionality to 0.01, i.e., after 100 ft the standard deviation is 1 ft. For a more rigorous treatment of odometric error, see [30].

3) Integration: Given (x_o, σ_o) and (x_m, σ_m), the position and standard deviation from odometry and matching, respectively, we can optimally combine these using the formulas [19], [3]

$$x_c = x_o + \frac{\sigma_o}{\sigma_o + \sigma_m}(x_m - x_o)$$

$$\frac{1}{\sigma_c^2} = \frac{1}{\sigma_o^2} + \frac{1}{\sigma_m^2}$$

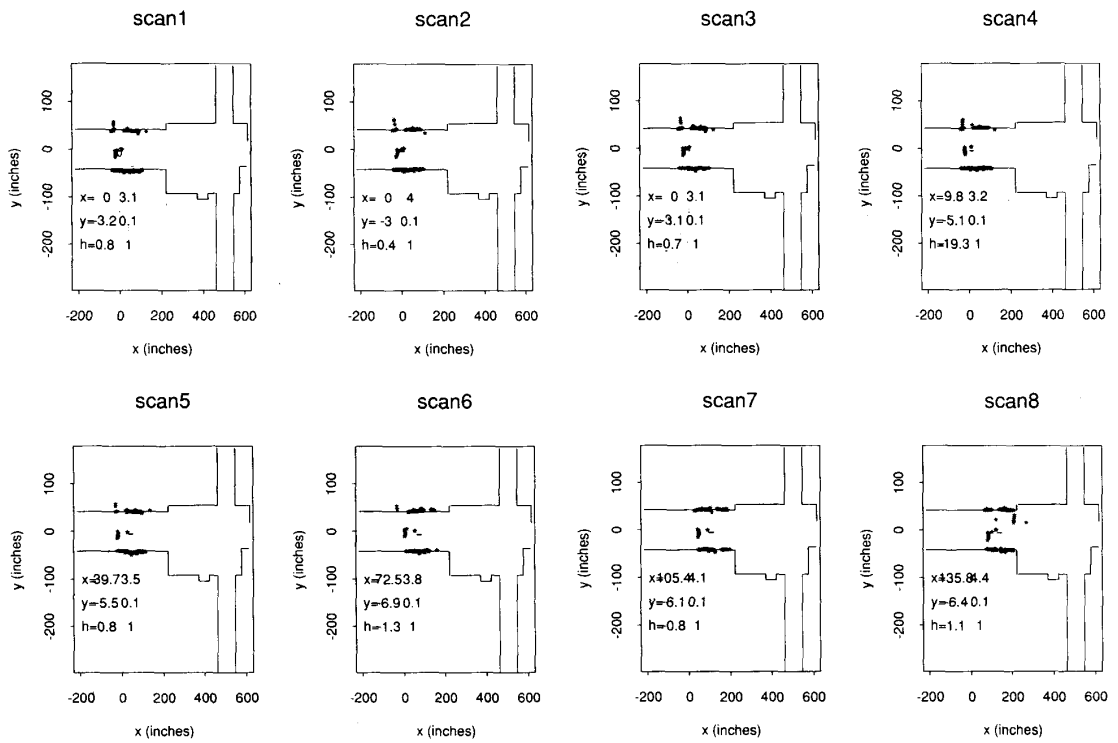
where (x_c, σ_c) are the updated values for the x position and its corresponding standard deviation. It is clear that if the standard deviation in the odometry is small compared with the matcher, the error term $(x_m - x_o)$ has little effect. Alternatively, if the standard deviation in the odometry is large compared with the matcher, almost all the error term is added to the corrected value. This is intuitively correct. In fact, for the case of the vehicle moving down a parallel corridor, very good estimates of the vehicle's orientation and position normal to the corridor can be made by the matcher, but the position of the vehicle along the corridor cannot be estimated by the matcher and its associated standard deviation is infinite. In these situations we rely on odometry until the matcher detects line features that are not parallel to the corridor.

This updated value is fed back to the odometry where it is used as the new value from which the current position is estimated. This is referred to as an *indirect feedback* configuration [19] in comparison to an *indirect feedforward* configuration, since the error correction is fed back into the odometry subsystem. In this way, the odometry errors are not allowed to grow without bound. Further, since the odometry estimate is corrected after each match or registration, failure of the rangefinder or matching subsystem does not lead to immediate failure of the robot system. Instead, the robot is free to continuing navigating for some period using only odometry until such time as the odometric estimates of the standard deviation in position exceed some unacceptable limits.

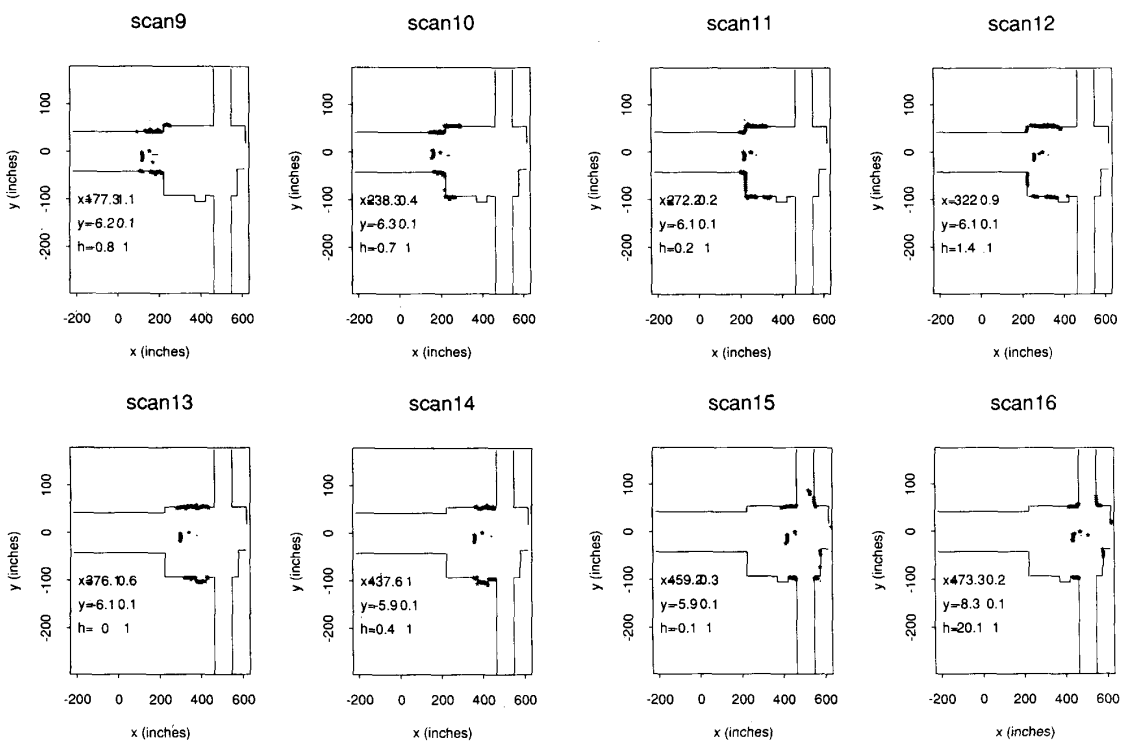
E. Experiments Results

Fig. 10 is a sequence of range images taken by the robot vehicle as it moved along a predetermined path. The vehicle is initially positioned at the origin (0, 0) and pointing in the positive x direction. The line segments illustrate the map of the environment, consisting of a corridor that opens up into an elevator bay, off of which are two additional corridors. Along the corridors are numerous doors that may be either open or closed, but are not modeled in the map. Data points obtained from the inside of rooms are therefore (hopefully) treated as spurious and discarded by the matcher. The pairs of numbers for x, y, h, represent the vehicle's estimate of its x, y position and orientation together with the associated standard deviations in these estimates.

At initialization the vehicle was given its initial position. In practice, the vehicle had been displaced approximately 3 in in its (x, y) position. Its orientation is approximately correct. The experimental setup lacked an independent position measurement system such as a real-time triangulation system. Consequently, the absolute accuracy of the vehicle could not be quantified except at its finish position. However, close examination of the path of the vehicle indicated that the mobile robot deviated from its path by no more that approximately 6 in. These deviations were not due to the low-level controller, which provided smooth trajectories. Small "lurches" in the vehicle's motion were visi-



(a)



(b)

Fig. 10. Sequence of range images as vehicle navigated along a corridor.

TABLE I
A SEQUENCE OF CORRECTIONS ESTIMATED BY THE MATCHER

Correction	x (inches)	y (inches)	θ (degrees)
1	0.0	2.8	0.7
2	0.0	0.1	−0.5
3	0.0	−0.1	0.3
4	0.0	0.1	−0.1
5	0.0	0.5	1.0
6	0.0	−0.2	−0.8
7	0.0	−0.2	−0.3
8	0.0	−0.1	0.3
9	−1.2	0.1	−0.6
10	6.7	−0.1	−0.5
11	−3.0	0.1	0.3

ble when the vehicle's position was updated by the matcher because, at these times the error between the commanded reference position and the estimated position would instantaneously become large (occasionally on the order of a few inches).

Table I tabulates the partial set of corrections made by the vehicle as it traverses its path. The first correction correctly determines the error in the y direction. A correction of 0.7° is also applied to the heading, but examination of the heading corrections suggests that this is noise. However, there is no correction in the x direction. This is because the vehicle cannot tell where it is along the corridor.[3] The matcher has an infinite standard deviation along the x direction. Only when the vehicle comes out into the elevator bay—see scan 9 of Fig. 10—is it able to begin to refine its x position.

Large collections of range points not associated with the model usually indicate people walking by. In addition, maintenance and cleaning personnel may occasionally leave unmodeled objects in the environment. However, almost all these data points are rejected as spurious because their vicinity to a modeled line segment is not within preset thresholds.

The relative precision of the odometry and matching estimates depends strongly on the particular data and on the frequency of matching. For a vehicle moving at 1 in/s with matching occurring every 10 s, the odometric standard deviation is approximately 1 in. Within the corridor, the matcher estimates the position of the vehicle *perpendicular* to the corridor with a standard deviation of approximately 0.1 in. Of course, the standard deviation along the corridor is infinite. For less extreme cases, such as the elevator bay (scans 9–11 of Fig. 10), the standard deviation in the x direction averaged 0.6 in.

The matcher's estimates of the standard deviations in (x, y, θ) assume a Gaussian noise form with zero mean. However, the prototype rangefinder appears to have systematic errors: range values have been seen to depend on the strength of the received signal. For example, very bright objects appear further away. We believe that the large corrections in the x direction at scans 9 through 11 are due in part to this problem. An improved rangefinder is currently under test that we hope will solve this problem. Other potential sources of systematic errors include errors in the map and errors in the alignment of the rangefinder relative to the vehicle. A misalignment of only 1° would cause an almost 2-in error in the y direction as the vehicle traveled 9 ft along the x direction.

[3] Remember that doorways are not modeled. Locally, the environment appears as two parallel lines.

V. SOFTWARE IMPLEMENTATION NOTES

Blanche has been primarily developed for use as an autonomous robot operating in a structured office or factory environment. It has also served as a testbed for experiments in real-time programming environments. In particular, we have used the C + + programming language [25] exclusively.

In a previous paper [5], we showed how the constructor mechanism associated with C + + classes can be used to guarantee the initialization and self-test of each subsystem of the robot. A complementary destructor mechanism can be used to guarantee safe termination of subsystems under most conditions. These mechanisms are completely transparent to both the user and his application program. Exception handling [6] is discussed, and it is shown how object-oriented programming facilities can be used to provide transparent recovery from subsystem failures during program execution, given some hardware redundancy. Most of the examples were demonstrated and tested using C + + running on Blanche.

I remain enthusiastic about the advantages of C + + for real-time robotics applications. In particular, data abstraction in the form of classes allows a modular code to be written that often has a one-to-one correspondence with underlying physical subsystems. The object-oriented programming facilities have also been exploited at the trajectory generation level where arc, line, and any other segment type are derived from a common base class. The low-level controller is unaware of the type of segment it is following, and this construction should allow for easy extensibility. Of course, C + +, like most languages, requires discipline on behalf of the user—I am certainly guilty of some dreadful fragments of C + + code, but it has also saved me from many run-time bugs.

VI. CONCLUSIONS

Trajectory generation and control procedures for enabling a robot cart to accurately and autonomously navigate around a structured environment, given a set of path data from an off-line path planner, have been described. These procedures have been implemented in an experimental robot cart. The on-board trajectory generator has a number of advantages including 1) it reduces greatly the amount of data needed from the path planner, so even a very low data rate, intermittent link to the cart is sufficient; 2) it provides a convenient separation between the path guidance and the cart control logic; and 3) it supports fail-safe operation if the vehicle's reference and measured states are continuously monitored.

The experience gained so far in this study indicates that the basic approach described above for guidance and control of a robot cart is sound. The separation of trajectory generation from the low-level cart controller provides a robust method for maintaining accurate tracking of any feasible path, once the controller unit has been properly adjusted to stabilize the closed-loop dynamics of the path control loop. We are still investigating the functional dependence of the control weights, C_1 and C_2 in (2), on speed. However, the current solution of scaling by the inverse of v^2 and v, respectively (where v is the speed of the vehicle) appears to be satisfactory.

The cart guidance system assumes accurate knowledge of the vehicle's position. Although passive or active beacons can provide a very accurate and cost-effective solution to position

estimation, the consequent need to modify the environment was considered undesirable and beacons were therefore not used. Instead, the position estimation system consists of 1) an *a priori* map of its environment, 2) a combination of odometry and optical range sensing, 3) an algorithm for matching the sensory data to the map, and 4) an algorithm to estimate the precision of the corresponding match/correction.

A 2D representation based on collections of line segments in the plane was chosen because 1) much of the robot's environment is uniform in the vertical direction; there is not much to be gained from a 3D representation, 2) the range sensor currently provides only 2D information (r, θ), 3) matching sensor data to 2D maps is significantly simpler than matching to 3D maps, and 4) pragmatically, a line segment description was chosen because an efficient and robust matching algorithm that used line segments had been developed.

The matching algorithm assumes that the displacement between image and model is small. This is a very reasonable assumption since odometry is providing the vehicle with a good estimate of position and matching occurs very frequently. Although the matching algorithm may be criticized for not being general, it does not have to be. Moreover, generality has been sacrificed for two very important characteristics, robustness and speed. Our experience is that while robustness is very desirable, it is very hard to achieve. The algorithm is intrinsically robust against incompleteness of the image, i.e., missing data points. Spurious data, e.g., people walking by, and incompleteness of the model are dealt with by deleting from consideration any points whose distance to their target segments exceed some limit. Note that, for the general matching problem, in which the assumption of small displacement is not necessarily valid, such an approach would not be possible.

The vehicle and associated algorithms have all been implemented and tested within a structured office environment. The entire autonomous vehicle is self-contained; all processing being performed onboard. We believe this vehicle is significant not just because of the sensing and algorithms described, but also because its implementation represents a very high level of performance at very low cost. There also appears to be a self-sustaining property to this configuration: Accurate knowledge of position allows for fast robust matching, which leads to accurate knowledge of position.

There are several areas of in which the vehicle might be improved. First, the need to provide a map of the environment can become tedious, and it would therefore be desirable to automate this step, e.g., the vehicle might build its own map [17]. Second, any map has so far been assumed to be perfectly accurate, all errors being considered due to sensor noise or errors in the matcher. In practice, the map also has an associated accuracy that should also be modeled. Third, the covariance matrix estimated by the matcher assumes there is no correlation between scans of the rangefinder. This is not necessarily true in practice because of 1) errors in the map and 2) sensor noise may be correlated with particular wall surfaces. Future work is directed to addressing these problems.

Acknowledgment

The development of Blanche would not have been possible without the help of many individuals. It is a pleasure to thank J. B. Kruskal and W. L. Nelson whose collaboration with the matching and control algorithms, respectively, were critical to the success of the project. Also thanks to G. L. Miller and E. R. Wagner for the optical ranger and D. A. Kapilow for the real-time computing environment, especially the debugging tools. Finally, R. A. Boie, W. J. Kropfl, J. E. Shopiro, F. W. Sinden, and G. T. Wilfong all provided help and assistance of one kind or another. Thank you.

References

[1] J. C. Alexander and J. H. Maddocks, "On the kinematics of wheeled mobile robots," *Int. J. Robotics Res.*, to be published.

[2] N. Ayache and O. Faugeras, "Building, registrating, and fusing noisy visual maps," in *Proc. Int. Conf. Computer Vision* (London), 1987, pp. 73–79.

[3] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. New York: Academic, 1988.

[4] I. J. Cox, "Blanche: An autonomous robot vehicle for structured environments," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1988, pp. 978–982.

[5] ——, "C + + language support for guaranteed initialization, failsafe termination and error recovery," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1988, pp. 641–643.

[6] I. J. Cox and N. H. Gehani, "Exception handling in robotics," *Computer*, vol. 22, no. 3, pp. 43–49, 1989.

[7] I. J. Cox and J. B. Kruskal, "On the congruence of noisy images to line segment models, in *Proc. Int. Conf. Computer Vision*, 1988.

[8] I. J. Cox, J. B. Kruskal, and D. A. Wallach, "Predicting and estimating the performance of a subpixel registration algorithm, unpublished, 1988.

[9] I. J. Cox and G. T. Wilfong, Eds., *Autonomous Robot Vehicles*. New York: Springer-Verlag, 1990.

[10] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE J. Robotics Automat.*, vol. RA-3, no. 3, pp. 249–265, 1987.

[11] A. Gelb, Ed. *Applied Optimal Estimation*. Cambridge, MA: MIT Press, 1974.

[12] G. Giralt, R. Chatila, and M. Vaisset, "An integrated navigation and motion control system for autonomous multisensory mobile robots," in *Proc. 1st Int. Symp. Robotics Res.* (Bretton Woods, NY) 1983, pp. 191–214.

[13] T. Hongo, H. Arakawa, G. Sugimoto, K. Tange, and Y. Yamamoto, "An automatic guidance system of a self-controlled vehicle," *IEEE Trans. Industrial Electron.*, vol. IE-34, no. 1, pp. 5–10, 1987.

[14] P. J. Huber, *Robust Statistics*. New York: Wiley, 1981.

[15] Y. Kanayama and B. I. Hartman, "Smooth local path planning for autonomous vehicles," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1989, pp. 1265–1270.

[16] D. A. Kapilow, "Real-time programming in a UNIX environment," in *Proc. Symp. Factory Automat. Robotics*, 1985, pp. 28–29.

[17] J. Leonard, H. Durrant-Whyte, and I. J. Cox, "Dynamic map building for an autonomous mobile robot," in *Proc. IEEE Int. Workshop Intell. Robots Syst.*, 1990, pp. 89–96.

[18] L. Matthies and S. A. Shafer, "Error modeling in stereo navigation," *IEEE J. Robotics Automat.*, vol. RA-3, no. 3, pp. 239–248, 1987.

[19] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, vol. 1. New York: Academic, 1979.

[20] G. L. Miller and E. R. Wagner, "An optical rangefinder for autonomous robot cart navigation," in *Proc. SPIE Mobile Robots II*, vol. 852, 1987, pp. 132–144.

[21] W. L. Nelson, "Continuous steering-function control of robot carts," *IEEE Trans. Industrial Electron.*, vol. 36, no. 3, pp. 330–337, 1989.

[22] ——, "Continuous-curvature paths for autonomous vehicles," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1989, pp. 1260–1264.

[23] W. L. Nelson and I. J. Cox, "Local path control of an autonomous vehicle," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1988, pp. 1504–1510.

[24] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.

[25] B. Stroustrup, *The C + + Programming Language*. Reading, MA: Addison-Wesley, 1986.

[26] D. J. Torrieri, "Statistical theory of passive location systems,"

IEEE Trans. Aerospace Electron. Syst., vol. AES-20, no. 2, pp. 183–198, 1984.

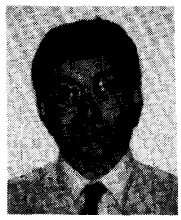
[27] T. Tsumura, "Survey of automated guided vehicle in Japanese factory," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1986, pp. 1329–1334.

[28] T. Tsumura, N. Fujiwara, and M. Hashimoto, "An experimental system for self-contained position and heading measurement of ground vehicle," in *Proc. Int. Conf. Advanced Robotics*, 1983, pp. 269–276.

[29] S. Uta *et al.*, "An implementation of michi-a locomotion command system for intelligent mobile robots," in *Proc. 15th ICAR*, 1985, pp. 127–134.

[30] C. M. Wang, "Location estimation and uncertainty analysis for mobile robots," in *Proc. IEEE Int. Conf. Robotics Automat.*, vol. 2, 1988, pp. 1230–1235.

[31] G. T. Wilfong, "Motion planning for an autonomous vehicle," in *Proc. IEEE Conf. Robotics Automat.*, 1988, pp. 529–533.



Ingemar J. Cox (S'79–M'83) received the B.Sc. degree in electrical engineering and computer science from University College, London, in 1980 and the D.Phil. degree in engineering science from the University of Oxford, in 1983.

He was a Principal Investigator with AT&T Bell Laboratories, Murray Hill, NJ, from 1984 to 1989. He is currently a Senior Research Scientist with the NEC Research Institute, Princeton, NJ. His research interests are broadly in the areas of autonomous robot vehicles, computer vision, and real-time programming environments. He coeditied (with G. T. Wilfong) the book *Autonomous Robot Vehicles* (Springer-Verlag).

Dr. Cox is a member of the ACM.