MARVEL: A System That Recognizes World Locations with Stereo Vision

David J. Braunegg

Abstract—MARVEL is a system that supports autonomous navigation by building and maintaining its own models of world locations and using these models and stereo vision input to recognize its location in the world and its position and orientation within that location. The system emphasizes the use of simple, easily derivable features for recognition, whose aggregate identifies a location, instead of complex features that also require recognition. MARVEL is designed to be robust with respect to input errors and to respond to a gradually changing world by updating its world location models. In over 1000 recognition tests using real-world data, MARVEL yielded a false negative rate under 10% with zero false positives.

I. INTRODUCTION

To navigate over a long lifetime, a mobile robot needs a memory of the world. To explore places of which it has no previous knowledge, a robot should be able to build its own world model. Due to the problem of cumulative error, exact metrical models of the world cannot be used [1]. The most promising alternative is a topological map that contains world locations and information about how they are connected [2], [3]. To use such a map, a robot needs the ability to recognize the locations contained therein.

The first part of the navigation support problem, then, is how to build models of world locations and use them for recognition. Because the world changes and the robot’s sensory input is imperfect, the second part of the problem is to maintain these models over time. The implemented system MARVEL (Model building And Recognition using Vision to Explore Locations) addresses these problems.

II. WORLD FEATURES FROM STEREO VISION

We use the Marr–Poggio–Grimson stereo algorithm to obtain the locations of world features [5]–[8]. This algorithm is based on Laplacian-of-Gaussian zero-crossing points that are associated with intensity edges in the images. Because such features usually correspond to physical edges, they characterize the distribution of objects in the visible world. We only use the long, near-vertical stereo features to model world locations for two reasons. First, we employ the heuristic that large objects (e.g., doorways, windows, bookcases) tend not to move and thus help to identify the specific areas in which they reside. Second, because our camera geometry uses horizontal epipolar lines, the localization of the stereo features deteriorates as the features become more horizontal. The long, near-vertical features are identified by first fitting straight lines to the 3-D edges using a method given by Pavlidis [9], then checking the lengths and slopes of the lines. Because the length constraint used was not severe, no attempt was made to aggregate short edges, which could occur due to occlusions, into longer edges.

To get a full view of the room containing the robot, we rotate the robot through 360°, taking overlapping views of the room [10]. The stereo features from these views are “pasted” together for a full representation of the room; the feature coordinates are converted into camera-centered world coordinates, and the ceiling and floor features are removed. The aggregate of these simple, easily derivable features are used to identify a location instead of relying on complex features that would themselves require recognition.

III. MODEL/ DATA REPRESENTATION

For a recognition system, the world is defined by what is observable by the sensors that are being used. Other researchers have investigated the use of full 3-D stereo features for map building [11], [12]. However, we wish to recognize rooms rather than navigate through them. To build the room representation for recognition, we project the vertical room features from the stereo algorithm to the groundplane (see Fig. 1). We have found that this representation sufficiently characterizes the room for the purpose of recognition. One implicit advantage of this 2-D representation is that partially occluded edges can be matched easily to their corresponding model edges.

We further abstract the data with an occupancy-grid representation of the room [13]. We impose a grid with 1-foot spacing on the floor of the room and mark each grid square containing an accepted stereo feature (see Fig. 2). The choice of the grid size depends on the stereo camera configuration, the sizes of rooms to be encountered, and the desired degree of localization. Although we use a sufficiently
large grid to avoid issues of stereo uncertainty, other researchers have explored explicitly modeling the uncertainty [14]-[16].

The final representation of a room is a set of grid squares that mark the locations of vertical room features. The grid is described in terms of a camera-centered coordinate system whose orientation is determined by the orientation of the robot when the first stereo pair is taken.

IV. RECOGNITION

For the recognition example below, the grid room representation shown in Fig. 2 is used as the initial model because MARVEL uses no a priori world knowledge. The representation obtained from another position and orientation in the same room serves as the data.

To recognize a data set with respect to a model, we use a least-squares algorithm to find the best fit of model to data and then evaluate that fit. This fitting process requires a good initial estimate of the transform (2-D translation and rotation) required to match the model and data. To obtain these initial estimates, we find transforms that align pairs of clusters of colinear model and data features. Other methods of grouping the points are possible [17].

For each initial alignment, a least squares process minimizes the distances between the model and data points by varying the translation (x, y) and rotation θ of the model. The search radius r used to determine model/data point correspondences is initially large, to pull in poor initial alignments, then reduced to find the best possible transformation (x, y, θ). This process is similar to the one described by Lowe [18]. However, given an alignment, we find all possible model/data point matches before performing the least-squares optimization. Initially, we tried refining the transformation by incrementally adding the matches to the least squares process, but found that one bad match could affect the rotation component of the transformation enough to generate a completely wrong final result.

Because 3-D edges, not surfaces, are obtained from the stereo algorithm, a visibility constraint cannot be enforced during the matching process. However, the long, vertical stereo features used are rarely occluded completely by surfaces in the environment; partially occluded edges can be matched easily due to the 2-D representations used.

The best recognition obtained from the set of initial guesses is the one that matches the most model points, with the transform variance used to break ties (see Fig. 3). The recognition also provides the robot's position and orientation with respect to the stored location model. Next, we determine if the model and data actually correspond to the same location. A set of data matches a model if the percentage of data points matched and the percentage of model points matched exceed previously selected thresholds. Both the model and data must be checked because neither is guaranteed to be error-free.

Matching the wrong model to the data (a false positive) is a serious error since the mobile robot uses this information to verify its location in its world map. Errorlessly deciding that the current data does not match any room model in the database (a false negative) is less serious. With a false negative, the current data is added to the database as a new room model, which can be merged later with the existing model for this room [4].

With a database of four different rooms, we ran over 1000 recognition tests with evolving models. (Three of the rooms have almost identical dimensions and were specifically chosen to provide worst-case tests for recognizing the correct room.) The recognition performance improved as the models were built up over time (see Fig. 4), with the false negative rate remaining below 10% for the fifth and subsequent models. No false positive recognitions occurred during testing. We verified that, in every case, MARVEL also determined the correct robot position and orientation within the recognized room.

Running on a Symbolics 3600 with a hardware convolver, the stereo algorithm took 1 min per image pair, with 48 s of that time devoted to edge finding. It took 84 s to check a data set against
a model, with most of the time consumed by the Symbolics array
utilities used to implement the least-squares minimization.

V. MODEL BUILDING

Once a recognition has been established, the model and data are
combined into a new, more reliable model. (If the current data does
not correspond to an existing model, it is entered into the database
as the model of a new location.) A weight is associated with each point
in the model and is updated based on the recognition. See Fig. 5(b)
for the updated model of Room 914 based on the recognition shown
in Fig. 3. The darker squares correspond to more heavily weighted
model points.

Combining new data with the current model serves two purposes.
First, new features that appear in the data are added to the model.
Second, the features that were used for recognition (and thus had
a model-data match) are given more importance in the recognition
process because of the weighted least-squares algorithm used.

VI. MODEL MAINTENANCE

By combining the model and data sets of a recognition into a
weighted model, we build the model up over time (see the Appendix).
The model update is based on the fact that the more often we
see something, the more confident we are of its existence. This is
important for three reasons. First, the world changes. Second, no
stereo vision algorithm (or any other sensing scheme) is perfect.
Third, the locations of world features determined by the stereo
algorithm (or, again, any other sensing scheme) are not perfect.

As data points are added over time, the model points approximate
a Gaussian distribution about the true location of the feature. Thus,
the model updates also serve to refine the locations of the world features
in the model. This approach avoids the additional computation needed
to model the uncertainty of the sensor data [19], [14].

A. The Model Point Weight Update

Even though the model should represent the importance of its
constituent points to recognition, the importance of any one point
must be limited. Otherwise, a point could become so entrenched in
a model that it might never be removed, even if current data repeatedly
confirms that its corresponding feature no longer exists.

The weight update function determines the rate at which model
point weights are updated based on the number of times that the points
were seen. Two different update functions (linear and exponential)
were used in testing MARVEL, with the former presented here. Given
a model \( M_{j-1} = \{ m_{i,j-1} \} \), which incorporates the information from
data sets \( D_1, D_2, \cdots, D_{j-1} \), the weight assigned to the \( i \)th model
point \( m_{i,j} \) in model \( M_{j} \), based on data set \( D_{j} \), is

\[
W_{m_{i,j}}(c; t_c) = \begin{cases} 
\frac{c}{t_c} & 0 \leq c < t_c \\
1 & t_c \leq c 
\end{cases} 
\]  

(1)

where \( t_c \) is the rise time (the number of times a point must be seen
to achieve full weight) and \( c \) is the number of times the point has
been seen in data sets \( D_1, D_2, \cdots, D_{j} \). Thus, a model point’s weight
is increased linearly each time it is used and decreased linearly each
time it is not. The weight of a point is constrained to a maximum value
of 1 and a point whose weight decreases to 0 is removed from the
model entirely. The response of the update function is changed by
varying the rise time.

Because occlusions and sensing errors are possible, the lowering
of a model point’s count should not happen immediately if it is not in the
current data. Instead, a hysteresis is imposed so that a point’s count
\( c \) is incremented by one each time the point is seen, but decremented
by one only if the point was last seen more than \( h \) data sets ago.

An age \( a \) for each model point indicates how many data sets ago
the point was last seen. Thus, a hysteresis of \( h = 1 \) indicates that a
model point must have been seen in the current or the immediately
preceding data set to not have its count and weight reduced. If \( h = 0 \)
and \( t_c = 1 \), the update replaces the model with the current set of data.

With each new data set, the information associated with each model
point must be updated. First, the age of the model point is either reset
or incremented depending on whether it was seen in the new data set:

\[
a_{m_{i,j}} = \begin{cases} 
0 & m_{i,j-1} \in D_{j} \\
1 & \text{otherwise}
\end{cases} 
\]  

(2)

The count of the number of times that the point was seen is then
updated, taking the age hysteresis into account:

\[
a_{m_{i,j}} = \begin{cases} 
\min(t_c, a_{m_{i,j-1}} + 1) & m_{i,j} \in D_{j} \\
a_{m_{i,j-1}} & m_{i,j} \not\in D_{j} \text{ and } a_{m_{i,j}} < h \\
a_{m_{i,j-1}} - 1 & m_{i,j} \not\in D_{j} \text{ and } a_{m_{i,j}} \geq h
\end{cases} 
\]  

(3)

Finally, the weight \( W_{m_{i,j}}(c; t_c) \) is determined by (1).

The two model update parameters \( t_c \) and \( h \) affect how the models
change over time. Larger rise times allow stable room features to play
a bigger role in the recognition process. Smaller rise times downplay
the importance of existing model points in favor of the incoming
location data. A large hysteresis provides a greater insensitivity to
data drop-outs due to occlusion, while a smaller hysteresis allows
quick response to changes in the world.

B. The Model Update Algorithm

An initial model simply consists of the data points first collected for
a particular room. The weight of a model point reflects its importance
to recognition, but no recognition has been performed using the
initial points. Thus, each initial model point weight is initialized to
\( W_{m_{i,j}}(1; t_c) \) because the point has only been seen once \( (a_{m_{i,j}} = 1) \).
The initial model for Room 914 is shown in Fig. 5(a) and is derived from
the data points shown in Fig. 2.

The model update algorithm uses a new set of data \( D_j \) and its
recognition with a model \( M_{j-1} \) to create a refined model \( M_j \). First,
the recognition transformation \((x, y, \theta)\) that brings \( M_{j-1} \) and \( D_j \) into
 correspondence is used to find the closest data point to each model
point. A model point is matched if its corresponding data point is
sufficiently close:

\[
m_{i,j-1} \in D_j \text{ if } \exists d_{k,j} \in D_j \text{ such that } \|m_{i,j-1} - d_{k,j}\| < \delta_{min}
\]

where the maximum allowable distance \( \delta_{min} \) is set to the smallest
search radius \( r \) of the least-squares transform refinement (see Section
IV). Next, the age of each model point is updated (2) based on
whether it was matched. The count and weight of each model point
are then updated according to (3) and (1), and points with a current
count of zero are removed from the model. Finally, any data point
that was not matched to a model point is added to the model with
weight \( W_{m_{i,j}}(1; t_c) \) and age \( a_{m_{i,j}} = 0 \).

Fig. 3 shows recognition of the initial model for Room 914 (see
Fig. 5(a)), with a new set of data. Fig. 5(b) shows the result of the
first update to the initial model. Comparing this new model with
the original, the added and updated model points are obvious. No points
have been retracted at this early stage in the model evolution. In later
model updates (see the Appendix), fading and deletion of old model
points can be seen as well.
VII. CONCLUSION
MARVEL is a working system that builds, maintains, and uses models of world locations for recognition to support navigation. We emphasize the use of simple, easily derivable features whose aggregate identifies a location, instead of complex features, which must themselves be recognized. Both the model builder and recognition system use this sparse real world data. The system not only recognizes the robot's location in the world, but also determines its position and orientation within that location. We reject a priori models in favor of models built by the robot itself, thus allowing the robot to explore new areas of the world. We accept the fact that any input data is noisy and explicitly provide a method for handling these data errors. We acknowledge that the world is not static and therefore provide a way for the location models to change over time as the perceived world changes. Finally, the recognition system has been tested on over 1000 model-data pairs from actual scenes with less than a 10% false negative recognition rate and a 0% false positive recognition rate.

APPENDIX
MODELS FOR ROOM 914
Fig. 5 shows the evolving models for Room 914 in the MIT Artificial Intelligence Laboratory. Model 1 was created from Data Set 1. For subsequent steps, Model \( n + 1 \) is the update of Model \( n \) based on recognition with Data Set \( n \), where each data set was acquired from a different position and orientation in the room. The darkness of a model point is proportional to its weight.
Note that the extraneous model point (due to an error in stereo matching) that was introduced in Model 2 (see Fig. 5(b)) is deleted in Model 6 (see Fig. 5(f)). This is an example of MARVEL's
ability to remove erroneous data from the models it builds. Also, compare the different locations of the features in the lower right corners of Models 8 (see Fig. 5(h)) and 12 (see Fig. 5(i)). These model features correspond to a large crate in one corner of the room that was moved during the testing of MARVEL. This demonstrates MARVEL's ability to respond to a changing environment by changing its location models.

REFERENCES

Position and Force Control for Constrained Manipulator Motion: Lyapunov’s Direct Method

Danwei Wang and N. Harris McClamroch

Abstract—A design procedure for simultaneous position and force control is developed, using Lyapunov’s direct method, for manipulators in contact with a rigid environment that can be described by holonomic constraints. Many manipulators that interact with their environment require taking into account the effects of these constraints in the control design. The forces of constraint play a critical role in constrained motion and are, along with displacements and velocities, to be regulated at specified values. Lyapunov’s direct method is used to develop a class of position and force feedback controllers. The conditions for gain selection demonstrate the importance of the constraints. Force feedback has been shown not to be mandatory for closed loop stabilization but it is useful in improving certain closed loop robustness properties.

I. INTRODUCTION

In order to use robot manipulators in many tasks, it is necessary to control both the position and velocity of the end-effector and the constraint force between the end-effector and the environment. Recent research has focused on simultaneous position and force control [2]–[5], [7]–[10], [13], [14]. Many control schemes have been proposed. Raibert and Craig have proposed a hybrid control method [9]; Yoshikawa has extended it to dynamic hybrid control [3] and Khatib has proposed an operational space formulation [13]. But in each of these papers, no explicit conditions were developed to guarantee closed loop stability. A careful stability analysis for such closed loop systems has only recently been given. McClamroch and Wang proposed a modified computed torque controller to achieve stable position and force tracking [5]. Mills and Goldenberg applied the theory for linear descriptor system to achieve stable position and force control [8]. McClamroch and Wang have recently developed conditions for local stabilization using a linear feedback controller [7]. These methods of [5], [7] require construction of a nonlinear coordinate transformation in which the constraints are trivial. This paper uses Lyapunov’s direct method to develop position and force control laws for constrained manipulators. This method overcomes the difficulties due to the nonlinearities of the robot dynamics and the coupling between the robot dynamics and the holonomic constraints. There is also no need to determine a nonlinear coordinate transformation as in [5], [8]. This approach represents an extension of Lyapunov’s direct method to constrained robot systems.

The objective of this paper is to demonstrate that Lyapunov’s direct method can form the basis for position and force control design and to present conditions that guarantee closed loop regulation. This paper also presents a case study to show that the control design method is easily applied to develop stabilizing controllers for constrained robot systems. We emphasize that the holonomic (equality) constraints are always assumed to be active; situations where constraints may be inactive are beyond the scope of this paper.

The organization of this paper is as follows. In Section II, constrained motion is modeled using a Lagrangian formulation and objectives are defined for position and force control. In Section III, Lyapunov’s direct method is used to develop position and force controllers; conditions for gain selection which guarantee closed loop stability are also provided. A pole assignment procedure is also developed. In Section IV, closed loop robustness properties are analyzed and discussed. In Section V, concluding remarks are made.

II. CONSTRAINED DYNAMICS AND CONTROL OBJECTIVES

The class of robot systems with holonomic constraints has a wide range of potential applications [4], such as a robot manipulator whose end-effector is always in contact with a constraint surface, multirobots holding a common object, etc. It has been argued that these constrained robot systems can be modeled using a Lagrangian formulation expressed by a set of differential-algebraic equations [2], [4].

Let \( q \in \mathbb{R}^m \) be a generalized configuration vector and \( \dot{q} \in \mathbb{R}^m \) be a generalized velocity vector. Suppose holonomic constraints on the motion are described by the following \( m \) algebraic equations

\[
\Phi(q) = 0
\]

where \( \Phi(q) = [\phi_1, \ldots, \phi_m] \) is at least twice differentiable. The kinetic and potential energy functions are denoted by \( K(q, \dot{q}) = 1/2 \dot{q}^T \mathbf{M}(q) \dot{q} \) and \( P(q) \); respectively, where \( \mathbf{M}: \mathbb{R}^m \to \mathbb{R}^{m \times m} \) is a symmetric positive definite definite matrix, and the potential energy function \( P: \mathbb{R}^m \to \mathbb{R} \) is at least twice differentiable. A Lagrangian function is defined for this constrained robot system as

\[
L(q, \dot{q}) = K(q, \dot{q}) - P(q)
\]

so that [2], [4]:

\[
\frac{d}{dt} \frac{\partial}{\partial q} L(q, \dot{q}) - \frac{\partial}{\partial q} L(q, \dot{q}) = J^T(q) \lambda + u
\]

where \( \lambda = (\lambda_1, \ldots, \lambda_m)^T \in \mathbb{R}^m \) is a vector of \( m \) constraint force and \( u \in \mathbb{R}^n \) is a vector of control inputs. \( J(q) \) is the Jacobian matrix of the constraint function \( \Phi(q) \). Using the definition of \( L(q, \dot{q}) \), the equations of constrained motion can be expressed as

\[
\mathbf{M}(q) \ddot{q} + \theta(q, \dot{q}) = J^T(q) \lambda + u
\]

where

\[
\theta(q, \dot{q}) = \frac{d}{dt} \frac{\partial}{\partial q} \mathbf{M}(q) \dot{q} - \frac{\partial}{\partial q} \frac{1}{2} \dot{q}^T \mathbf{M}(q) \dot{q} + \frac{\partial}{\partial q} P(q).
\]