Vision-Based Control of Mobile Robots

Darius Burschka and Gregory Hager
Computational Interaction and Robotics Laboratory
Johns Hopkins University
Baltimore, MD 21218
{burschka|hager}@cs.jhu.edu

Abstract

This paper presents an approach for direct control of a mobile robot to keep it on a pre-taught path based solely on the perception from a monocular CCD camera. In particular, we present a novel vision-based control algorithm for mobile systems equipped with a conventional camera and a pan-tilt head or with an omnidirectional camera. This algorithm avoids numerical instabilities of previously reported approaches. The experimental performance of the method as well as its practical limitations are discussed.

1 Introduction

A common problem in most applications of mobile systems is to keep the vehicle on a pre-defined path. This path may be a corridor through a factory to transport parts from one machine to another, or a route through a building to give a pre-specified tour [3], or it may be a known path between offices in case of a courier robot [5]. Several systems have been proposed to solve this problem, most of which operate based on maps [4, 10, 11, 14, 13] or based on localization from artificial landmarks in the environment [8, 5].

In many cases it is not necessary to use sophisticated map-based systems to control the paths of the robot—instead a simple teaching phase may be sufficient to specify it. This is in fact our goal: to develop a system that can be walked in a teaching step through the environment. During this teaching phase it learns the path based on the sensor perception and can then later robustly repeat this path using the same sensing together with previously stored information.

One solution to this problem is to simply use dead-reckoning to keep track of the position in the world [12]. As is well known, accuracy that can be achieved is highly dependent on the accuracy of prior calibration and the quality of the surfaces on which the system operates. In general, such approaches will, over time, accumulate error and must be corrected by an external reference.

Map-based systems use a stored two or three-dimensional representation of the environment together with sensing to provide such a reference. However, it is not clear that building a metrically accurate map is in fact necessary for navigation tasks which only involve following the same path continuously. Another approach would be to use no prior information, but rather to generate the control signals directly from only currently sensed data. In this case no path specification at all is possible.

Our approach can be seen as a compromise between these two extremes, where a video camera is used as a passive sensing device for both recording and replaying a path. In particular, this article discusses a novel formulation of the vision-based control problem for path replaying which avoids the limitations of a previously reported method [6].

The remainder of this article is structured as follows. The next section describes the geometry of the vision-based control problem for conventional cameras equipped with a pan-tilt head. Section 3 describes the system model of the non-holonomic platform of the mobile system, and describes the path recording and control methods we use. It also extends the ideas to an omnidirectional camera system. Section 4 presents experimental results from the system. We close with a discussion of future work.

2 Navigation in a Local Segment

A conventional camera with a focal length of 8mm covers only a narrow field — approximately 50° in horizontal and vertical directions. In most cases a pan-tilt head must be used to compensate for the rotation of the mobile base and keep visual features within the field of view. In [6] a modification to a control algorithm based on a planar image plane was proposed to permit application of the algorithms derived for a
static camera to a camera mounted on a pan-tilt head. It was subsequently found that this approach has the drawback of yielding a numerically ill-conditioned system of equations for large pan angles. In this section, we reformulate the vision-based control problem in terms of a spherical projection that avoids these difficulties.

2.1 System Model

We assume throughout the article a non-holonomic mobile system with kinematics similar to a unicycle. The system operates in the x-z plane and rotates around the y-axis (fig. 1). Let the Cartesian coordinates \((x_g, z_g)\) specify the robot position in space and \(\Theta_g\) describe its orientation. The motion of the robot can be described by its forward velocity \(v\) and the superimposed rotational velocity \(\omega\) as \(s = (v + \omega)T\).

![Figure 1: Coordinate system used in the system.](image)

Given these definitions, the kinematics of the system can be described in the Cartesian coordinate system as

\[
\begin{align*}
x_g &= v \cdot \cos \Theta_g \\
z_g &= v \cdot \sin \Theta_g \\
\Theta_g &= \omega
\end{align*}
\]

2.2 Spherical Image Projection

We employ cylindrical coordinates for points observed by a system moving in the plane of the floor. Let us assume the origin of the camera is at the center of rotation of the mobile system, the optical z axis points in the “forward” direction of motion of the robot, and the x axis points to the “right” of the direction of forward travel (fig. 1).

A point in space relative to the robot can then be described by the triple \((r_i, \varphi_i, y_i)\). The polar coordinates \(r_i\) and \(\varphi_i\) describe the position of the projection of the point onto the horizontal plane containing the camera optical axis. The vertical distance of the point from this plane is given by \(y_i\).

We can also define the elevation angle \(\gamma_i\) above that plane as

\[
\gamma_i = \arctan \frac{y_i}{r_i}
\]

The pair \((\gamma_i, \varphi_i)\) are the spherical coordinates of the projection of the point located at \((r_i, \varphi_i, y_i)\) into the (spherical) camera image. Note that \(\varphi_i\) increases clockwise, and \(\gamma_i\) increases in the downward direction.

3 Vision-Based Control

We compute feature observations by first normalizing for pixel spread and focal length (that is, converting to a unit focal length camera with the feature image coordinates \((u_i, v_i)\)), and then computing observed angles in the camera image as

\[
\begin{align*}
\alpha_i &= \arctan u_i = \arctan \frac{x_i}{z_i} \\
\beta_i &= \arctan \frac{v_i}{\sqrt{x_i^2 + z_i^2}} = \arctan \frac{y_i}{\sqrt{x_i^2 + z_i^2}}
\end{align*}
\]

For the imaging system consisting of a regular camera mounted on a pan-tilt head, the angles calculated in the camera image, \((\alpha_i, \beta_i)\) and the angles of the incident light rays \((\varphi_i, \gamma_i)\) are related by:

\[
\alpha_i = \varphi_i + \theta_{pl} , \quad \beta_i = \gamma_i
\]

where \(\theta_{pl}\) is the azimuth angle of the pan-tilt head. The necessity for the distinction between these two quantities will become clear in the case of an omnidirectional camera (sec. 3.4).

3.1 The Image Jacobian

Now, assuming holonomic motion in the plane, we can compute the following image Jacobian relating the change of angles in the image, \((\alpha_i, \beta_i)\), due to changes in motion in the plane, \((x_i, z_i)\) from the eq. 3:

\[
\mathcal{J}^t = \begin{pmatrix}
\frac{\partial \alpha_i}{\partial x_i} & \frac{\partial \alpha_i}{\partial z_i} & \frac{\partial \alpha_i}{\partial \Theta_g} \\
\frac{\partial \beta_i}{\partial x_i} & \frac{\partial \beta_i}{\partial z_i} & \frac{\partial \beta_i}{\partial \Theta_g} \\
-x_i \frac{x_i}{x_i^2 + z_i^2} & -x_i \frac{z_i}{x_i^2 + z_i^2} & -1 \\
\frac{y_i}{(x_i^2 + y_i^2 + z_i^2)^{3/2}} & \frac{y_i}{(x_i^2 + y_i^2 + z_i^2)^{3/2}} & 0
\end{pmatrix}
\]

1708
The dependency on the Cartesian coordinates can be avoided considering the geometry of the system to:

\[ R_i = \sqrt{x_i^2 + y_i^2 + z_i^2} = \frac{y_i}{\sin \beta_i} \]

\[ r_i = \sqrt{x_i^2 + z_i^2} = \frac{y_i}{\tan \beta_i} \]

\[ J_i^T = \begin{pmatrix} \frac{x_i}{r_i} & -\frac{x_i}{r_i} & -1 \\ \frac{y_i}{R_i^2 r_i} & -\frac{y_i z_i}{R_i^2 r_i} & 0 \\ \frac{z_i}{R_i^2 r_i} & -\frac{z_i y_i}{R_i^2 r_i} & -1 \\ \frac{\tan \beta_i \cos \varphi_i}{y_i} & \frac{\tan \beta_i \sin \varphi_i}{y_i} & 0 \end{pmatrix} \]  

Note in particular that the image Jacobian is a function of only one unobserved parameter, \( y_i \), the height of the observed point. Furthermore, this value is constant for motion in the plane. Thus, instead of estimating a time-changing quantity as is the case in most vision-based control, we only need to solve a simpler static estimation problem.

3.2 Estimating Parameters for the Jacobian Matrix

We can estimate the value \( y_i \) for each tracked object \( O_i \) from the information collected during the teaching phase of the system. Consider two different robot positions (labeled \( p_1 \) and \( p_2 \) in fig. 2) in space with coordinates \((x_1, z_1, \Theta_1)\) and \((x_2, z_2, \Theta_2)\) in a distance \( d \neq 0 \) from each other. For simplicity, we take the first location to be the origin. Then we can formulate the following equations based on fig. 2 and eq. 4:

\[ \alpha_j' = \alpha_j + \Theta_j \]

\[ \tau = \alpha_2' - \alpha_1' \]  

The orientation of the resulting triangle depends on the direction \( \psi \) in which the vehicle moved in absolute coordinates. Since we have chosen the first position as the origin, this is simply

\[ \psi = \tan 2 \left( \frac{p_2}{z_2} \right) \]

The knowledge of \( \psi \) allows us to calculate the other angle \( \nu \) of the triangle as

\[ \nu = \alpha_1' - \psi \]

Based on these expressions we can calculate the distance \( d_2 \) to the tracked object \( O_i \) from the position \( p_2 \) to be

\[ d_2 = \frac{\sin \nu}{\sin \tau}, \quad \tau \neq 0 \]

Equation 11 is valid only for \( \tau \neq 0 \), which is true for all landmarks lying outside of the vertical line going through the center point of the image (\( \alpha = 0 \)). Calculation of \( d_2 \) in this case is not possible because no horizontal displacement can be detected.

The estimated value for \( d_2 \) is used in eq. 2 together with the current vertical angular position \( \beta_i \) of the landmark \( O_i \) to estimate \( y_i \) to be

\[ y_i = d_2 \tan \beta_i = d \cdot \frac{\sin \nu}{\sin \tau} \tan \beta_i \]

The resulting value is used in the replay mode to calculate the inverse of the Jacobian matrix (eq. 5). Although formulated for a single motion, better results are obtained by formulating this system for every system motion and solving a least-squares problem.

3.3 Processing Sequence

A typical application of the system is subdivided into two phases: teaching phase and replay phase.

Teaching Phase In our current implementation the system starts with a selection of the landmarks, which are used in the further processing. This selection can be done manually by the user, who selects rectangular regions in the image, or based on an automatic landmark selection described in [9]. Once the landmarks are selected SSD trackers from the XVision library [7] are used to track their position in consecutive image frames.

The position, \( p_j^t \), of the tracked patterns (in the following text just referred to as trackers) is used to calculate the center \( C^j \) of the centroid between them.
\[ C^t = \frac{1}{N} \sum_{i=1}^{n} p_i^t \]  

(13)

The horizontal deviation \( \Delta C^t \) from the center of the image is used as an error signal in the control algorithm moving the pan-tilt head. In this way the system tries to keep the trackers as far away from the image boundary as possible.

The system waits until it detects an initial movement and then it starts the recording of the path. Images \( T^t \) are acquired from a monocular camera in equidistant time intervals \( \Delta T \) and for each frame the position of the trackers in the image \( p_i^t \) is estimated. These positions are stored in a matrix \( M_p \), which is updated in each step \( t \in 1, \ldots, K \).

\[
M_p = \begin{pmatrix}
   p_1^1 & p_1^2 & \cdots & p_1^K \\
p_2^1 & p_2^2 & \cdots & p_2^K \\
\vdots & \vdots & \ddots & \vdots \\
p_N^1 & p_N^2 & \cdots & p_N^K
\end{pmatrix}, \quad p_i^t = \begin{pmatrix}
   \alpha_i \\
   \beta_i
\end{pmatrix} \quad (14)
\]

At the end of the teaching phase, which in our system is recognized as a stop, the estimation step is performed (eq. 12) to calculate the values \( y_i \) for all trackers.

**Replay Phase** In the replay phase the values \( M_p[t][\hat{i}] \) representing the stored positions for the tracker \( i \) at the timestamp \( t \) together with the estimations of \( y_i \) are used to calculate \( \partial x, \partial z, \partial \phi \)

\[
\Delta p_i^t = M_p[t][\hat{i}] - p_i^t \\
\begin{pmatrix}
   \partial x \\
   \partial z \\
   \partial \phi
\end{pmatrix} = K \cdot (J^t)^{-1} \cdot \Delta p_i^t \quad (15)
\]

The constant vector \( K \) is chosen to correctly represent the length of the time interval \( \Delta T \) between the calculation of control inputs relative to the recording time interval.

The problem of the ambiguity in the position while using just one tracker can be avoided by combining responses from several trackers in a single Jacobian matrix. For this to happen the Jacobian matrix \( J^t_h \) estimated in eq. 5 is "stacked" to find the correction \( (dx, d\phi) \), which would best approximate the observed errors in the image \( \Delta p_i^t \) based on the responses from \( p \) trackers.

\[
J^t = \begin{pmatrix}
   J_1^t \\
   \vdots \\
   J_p^t
\end{pmatrix} \Rightarrow (J^t)^{-1} = (J^T J^t)^{-1} J^T
\]

(16)

The information permits us to generate the control signals for the non-holonomic robot assuming movements on circle segments or straight lines, which are a special case of a circle with radius \( r \to \infty \). If the actual position \( P_1 \) of the robot with the orientation \( \Theta_1 \) differs from the required position \( P_2 \) a correction movement on a circle around the point \( O \) is generated (fig. 3).

![Correction of the positional error with a movement on a circle with the radius \( r \)](image)

In point \( P_1 \) the robot has the orientation \( \Theta_1 \). The question is, how to construct a circle connecting \( P_1, P_2 \) with \( \Theta_1 = 0 \) as tangential direction in \( P_1 \). The radius \( r \) of this circle \( PO \) lies on the line with the orientation \( \beta = \frac{\pi}{2} \) in the local coordinates of the robot. Therefore, the point \( O \) must fulfill the following equations:

\[
r = \frac{||P_1 O||}{2 \cdot \sin \alpha}, \quad \alpha = \arctan(\frac{dx}{dz}) \quad (17)
\]

The value \( r \) is the radius of the reconstructed circle and \( \sqrt{dx^2 + dz^2} \) is the displacement between the actual and ideal positions. In most cases the angle \( \tau \neq 90^\circ \), which means that for larger displacement errors the orientational error of the system cannot be corrected. Therefore, the two control velocities for the robot are calculated as

\[
v = v_r + v_c, \quad v_c = |\sqrt{dx^2 + dz^2}|/T \\
\omega = \begin{cases} 
   \omega_r + \frac{\pi}{T} \frac{dx}{dz}, & |\sqrt{dx^2 + dz^2}| > 0.05 \|v\| \\
   \omega_r + \frac{\pi}{T} \frac{dx}{dz}, & \text{otherwise}
\end{cases} \quad (18)
\]

where \( \omega_r, v_r \) are the pre-recorded velocities from the teaching phase, in case a feed-forward mode is chosen.

We note that the order of the columns in the matrix \( M_p \) can flipped for each replay to allow bidirectional
control on the pre-taught path. This makes it simple to perform repeated trials without reinitializing the system.

3.4 Extension to an Omnidirectional Camera System

The algorithm described above uses the angular representation of the imaged objects $p_i$. This representation is not necessarily bound to the planar image plane of the conventional camera, but it can be used with any geometry of the image plane as long as a single viewpoint $F$ in the projection is ensured. In this case the incoming optical rays intersect in a single point and allow a simple mapping of the pixel position to an elevation angle $\beta_i$. This can be achieved with a hyperboloid mirror and a standard perspective camera in an omnidirectional camera system [1]. For our algorithm the angle $\beta_i$ in $p_i^e$ is replaced by the angle $\gamma_i$ for the pseudo-camera with the focal point in $F$.

\[ p_i^e = \begin{pmatrix} \alpha_i \\ \gamma_i \end{pmatrix} \] (eq. 23)

Further modifications in the tracking algorithms, which need to take into account the distortions of the tracked patterns due to the mirror geometry (fig. 5), are straightforward to implement.

![Figure 5: Planar model: (left) original omnidirectional image, (right) unwarpd planar reconstruction of a partial view.](image)

The first approach is to compute a perspective camera view from the omnidirectional image and to use this image as input to the existing tracking algorithms. The current implementation allows computation of a 60x60 patch in the image with 200 Hz update rate on a PentiumIII running at 1 GHz, which allows real time tracking with omnidirectional cameras.

4 Results

4.1 Quality of the y-Estimation

Figure 6 shows the positions of the tracked patterns in an example scene at the beginning and at the end of the teaching phase.

![Figure 6: Tracking of selected patterns in an example scene.](image)

The results of $y$ estimation (in meters) for the tracked patterns on straight paths with different view angles to the tracked features are shown in the following table.
<table>
<thead>
<tr>
<th>distance travelled</th>
<th>Estimated $y_i$ in [m]</th>
<th>feat1</th>
<th>feat2</th>
<th>feat3</th>
<th>feat4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3m</td>
<td>-0.9319</td>
<td>-0.8814</td>
<td>0.786</td>
<td>-0.1749</td>
<td></td>
</tr>
<tr>
<td>2.1m</td>
<td>-0.9563</td>
<td>-0.8087</td>
<td>0.7175</td>
<td>-0.2433</td>
<td></td>
</tr>
<tr>
<td>1.8m</td>
<td>-0.9685</td>
<td>-0.8032</td>
<td>0.7576</td>
<td>-0.2594</td>
<td></td>
</tr>
<tr>
<td>0.33m</td>
<td>-1.0436</td>
<td>-0.8477</td>
<td>0.7229</td>
<td>-0.1271</td>
<td></td>
</tr>
<tr>
<td>0.4m</td>
<td>-1.2560</td>
<td>-0.8393</td>
<td>0.7814</td>
<td>-0.1274</td>
<td></td>
</tr>
<tr>
<td>0.45m</td>
<td>-1.3476</td>
<td>-0.8927</td>
<td>0.8047</td>
<td>-0.1544</td>
<td></td>
</tr>
</tbody>
</table>

The estimation from the shorter paths in the lower part of the table is, as expected, more inaccurate and sensitive to the view angle of the camera. The inaccuracy in the calibration and the limited resolution of the image deteriorate the achievable results. The tracked objects were initially 7m from the camera. An example of estimates for an object in an initial distance of 2m is shown in the following table.

<table>
<thead>
<tr>
<th>distance travelled</th>
<th>$y_i$ in [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.65m</td>
<td>-0.4950</td>
</tr>
<tr>
<td>0.5m</td>
<td>-0.4922</td>
</tr>
<tr>
<td>0.26m</td>
<td>-0.4942</td>
</tr>
<tr>
<td>0.3m</td>
<td>-0.4805</td>
</tr>
</tbody>
</table>

The estimates on curved paths have a larger deviations in the estimated values. The traveled distances in this case are the distances between the endpoints of the path.

<table>
<thead>
<tr>
<th>distance travelled</th>
<th>Estimated $y_i$ in [m]</th>
<th>feat1</th>
<th>feat2</th>
<th>feat3</th>
<th>feat4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2m</td>
<td>-0.8542</td>
<td>-0.9123</td>
<td>0.6789</td>
<td>-0.1223</td>
<td></td>
</tr>
<tr>
<td>1.9m</td>
<td>-1.0834</td>
<td>-0.9347</td>
<td>0.6922</td>
<td>-0.1367</td>
<td></td>
</tr>
<tr>
<td>0.33m</td>
<td>-1.4012</td>
<td>-0.9786</td>
<td>0.5089</td>
<td>-0.1578</td>
<td></td>
</tr>
</tbody>
</table>

Some of the differences in the above measurements result from slightly different region selection in consecutive measurements. In each case a 32x32 pixels region is tracked, but the exact selection of the same clipping region was difficult.

In any case this estimation is based on the odometry of the vehicle and, therefore, all paths involving rotations deteriorate the estimation results. The best results can be achieved on straight path segments. The estimation algorithm should be extended to evaluate the path looking for such segments before starting the estimation.

### 4.2 Path Following Results

Visual based control systems are designed to correct the error signal of the input to zero. In our case the position of the tracked objects $Q_i$ in the teaching and the replay phase should remain the same, which is independent from the internal camera parameters. That means that incorrect estimates of the focal length $f$ in the camera system have only an effect on the absolute values of the derived control signals, but the system still correctly recognize the zero error state in the image. This advantage is lost in case of the combination of a conventional camera system and a pan-tilt head.

The reason is the coupling of the pan-tilt horizontal angle in the calculation of the feature position in eq. 4. Incorrectly estimating $f$ results in an error in the estimated error angle $\Delta C^e$ as described in section 3.3, which in turn leads to incorrect positioning of the head following the tracked objects and to an incorrect value of $\alpha^e$.

These problems do not occur in the omnidirectional systems, because in this case both angles responsible for the generation of the control signals are directly mapped on the specific area of the camera image (eq. 22).

In our experiments the path lengths were limited to 3.5m direct distance between start and endpoint. On these paths the vehicle remained within 5cm of the path on straight paths with an endpoint deviation of 6cm. With a disabled pan-tilt correction these results improved to below 2cm path deviation with endpoint deviation of 4cm. The bulk of the error seems to lie in the calibration problem described above.

In the experiments the vehicle was shifted from the correct endpoint of the path at about 50cm with an additional rotation of approx. 40° and was able to return to the pre-taught path. Tests with larger displacements will follow to check the actual limits of the algorithm. It seems that the true problem with larger shifts is the new path followed by the vehicle to reach the old path. This may result in collisions with obstacles, which were avoided in the teaching phase.

### 4.3 Stability of the System

In our experiments, we used a proportional controller to generate the control signals for the system. The whole computation was performed on a single Pentium@233MHz with a cycle time of 100ms. This slow cycle time in terms of the control theory did not allow to use high gains in the controller, which may cause oscillations. This resulted in an observed lag between the actual and desired position of the vehicle.

We observed a deviation of approx. $\Delta = 10cm$ (fig. 7) on curves with a radius of approx. 1m. This deviation was mainly caused by the difference between the desired position of the recorded point and the actual position of the vehicle. This deviation should not
be confused with the path deviation mentioned in section 4.2 in which the location of the vehicle scattered. This was a constant deviation due to the controller design.

![Path Deviation](image.png)

Figure 7: Deviation $\Delta$ from the desired path (dashed line) due to the lag of the used controller.

An improvement of the path control was achieved by replacement of the P-controller with a PD-controller for the control of the rotational speed $\omega$ with better responses to the demands of the controlling algorithm. This made it possible to keep the vehicle within the path deviation mentioned in section 4.2.

5 Conclusion and Future Work

Our work is part of a system for sensor based navigation in which natural landmarks are used as markers or landmarks. In this system a set of natural landmarks is used for navigation in a given segment of the path. The whole path of the system is subdivided into segments in which a given set is visible. The sets change each time any of the landmarks disappear due to occlusions or limited field of view of the conventional camera system. This defines the boundaries between the path segments. The landmarks can be selected manually by the user in a teaching phase or they are selected automatically by the system.

We have discussed the portion of the system that is responsible for the robust data acquisition during the teaching phase and for the control of the vehicle to stay on a pre-defined path during the replay phase in a given path segment. The experimental validation on our system proves the usefulness of the presented approach for vision based navigation in indoor environments.

The planned extension of the system is the usage of an omnidirectional camera system instead of the combination of the camera and the pan-tilt unit. This will allow to increase the length of the path segments in which a given set of landmarks is visible, because of the $360^\circ$ field of view on such systems. The necessity for exact calibration will also be unnecessary as already explained in the results section. Another useful extension is the addition of a compensation for vertical swings of the camera, which occur sometimes on the three-wheel system and lead to an incorrect estimation of the parameters $(\dot{x}, \dot{y}, \dot{\theta})$.

Acknowledgments

This work was supported by DARPA under DAAE07-98-C-L031 and the MARS project, and by the NSF RHA program.

References