Visual Odometry and Control for an Omnidirectional Mobile Robot with a Downward-Facing Camera

Marc Killpack[1], Travis Deyle[1], Cressel Anderson[1], and Charles C. Kemp[1]

[1]Healthcare Robotics Lab, Georgia Institute of Technology, USA

***Abstract*— An omnidirectional Mecanum base allows for more flexible mobile manipulation. However, slipping of the Mecanum wheels results in poor dead-reckoning estimates from wheel encoders, limiting the accuracy and overall utility of this type of base. We present a system with a downward-facing camera and light ring to provide robust visual odometry estimates. We mounted the system under the robot which allows it to operate in conditions such as large crowds or low ambient lighting. We demonstrate that the visual odometry estimates are sufficient to generate closed-loop PID (Proportional Integral Derivative) and LQR (Linear Quadratic Regulator) controllers for motion control in three different scenarios: waypoint tracking, small disturbance rejection, and sideways motion. We report quantitative measurements that demonstrate superior control performance when using visual odometry compared to wheel encoders. Finally, we show that this system provides high-fidelity odometry estimates and is able to compensate for wheel slip on a four-wheeled omnidirectional mobile robot base.**

I. Introduction and Related Work

Omnidirectional robot bases have been developed for research purposes since the 1980s and are rising in popularity for mobile manipulation. One such base is the Mecanum or Swedish wheel omnidirectional base where each wheel consists of a number of passive rollers mounted at a fixed angle to provide off-axis forces [1]. Our robot, Cody, with a Mecanum base is shown in Figure 1. We have found that wheel slip, which is especially common for side-to-side translation and rotation of the mobile base, results in very poor wheel-encoder odometry estimates. The degraded odometry and dead reckoning navigation is especially problematic for motion controllers, and thus, for mobile manipulation tasks that require motion estimates.

Overconstrained kinematic equations exist to detect slip, but do not quantify it [1]. This problem is exacerbated by other factors such as unbalanced wheel friction caused by uneven weight distribution. To successfully and effectively control the motion of the robot to facilitate manipulation tasks, we propose an additional form of odometry.

The cost and ubiquity of cameras has made them a promising supplement to wheel encoder odometry, particularly for conditions where wheels commonly slip on mobile robot bases. Work in visual odometry (VO) with both monocular and stereo sequences of images has shown improvement over simple dead-reckoning from wheel encoders in these situations [2], [3], [4]. Other researchers have examined inexpensive optical flow sensors, like those used in optical mice, but this work used extensive calibration or did not subsequently implement mobile base controllers for evaluation [5], [6], [7], [8]. Work by Cooney *et al* probably most resembles our own [9]. They employed optical mice and a PID controller to command a Mecanum base. However, they show the results of only one trajectory and ignore the fact that the system is actually a MIMO system and not theoretically well suited to SISO control schemes.

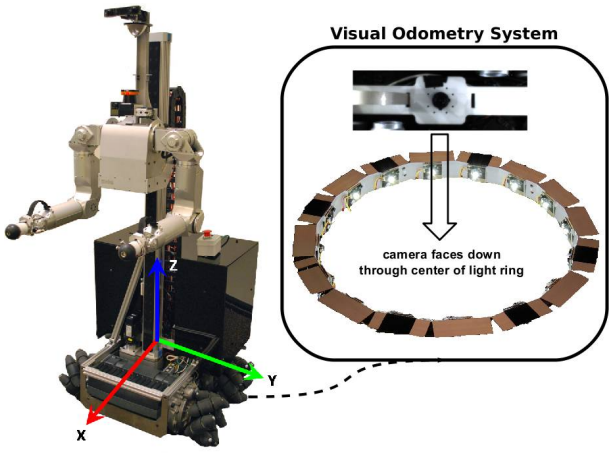

Fig. 1. A mobile manipulator platform with superimposed coordinate frame. The downward-facing camera and light ring sit under the robot to provide visual odometry.

By constraining ourselves to planar indoor environments common for mobile service robots, VO techniques can be simplified as robot pose is reduced from six degrees-of-freedom (6-DoF) to a planar rotation and translation (3-DoF) [10]. A calibrated downward-facing camera (shown in Figure 1) captures images of the floor. We use a homography to adjust for slight camera mounting offset angles, thereby rectifying (planarizing) the image. We mounted the camera on the robot's underside and included a specialized light which allows us to maintain a controlled environment in terms of lighting and specular reflections. Our setup results in robust VO estimates even when the robot operates in challenging environments with moving objects or people and in variable or low light conditions. We also believe this technique may facilitate future research in localization and mapping similar to work by Kelly [11], but applied to homes and office work spaces.

Our work is distinguishing in a number of ways. First, we demonstrate a closed-loop LQR controller on a Mecanum

base with downward-facing camera. Others have examined PID controllers on Mecanum bases using optical flow [9], but we report experiments that provide more extensive quantitative evaluation of both the PID controller and the new LQR controller. Second, we demonstrate performance of a visual odometry system whose closed-loop error is less than 0.52% over short distances with significant angular rotations, which is comparable to other results reported in the literature. We show that the visual odometry estimates are sufficient to construct robust closed-loop controllers that perform significantly better than controllers using the wheel encoder odometry and are capable of operation in challenging or adverse conditions, such as in large crowds or low ambient lighting. Finally we have already used our system in application for mobile manipulation tasks, [12].

II. The Visual Odometry System

A. The Mobile Robot Platform

The mobile robot, Cody, shown in Figure 1 is a statically stable mobile manipulator with arms from MEKA Robotics (MEKA A1), an omnidirectional mobile base from Segway (RMP 50 Omni composed of two RMP 50 differential drives) and a 1-DoF linear actuator from Festo. The Mecanum wheels are eight inches in diameter.

Two Mac Mini computers running Ubuntu GNU/Linux provide on-board computation. One computes visual odometry and has Intel IPP installed to speed up image processing. The other computer logs odometry data and transmits control commands to the Segway base. The Segway base operates as two differential drive controllers (front and back), each taking a forward linear velocity and an angular velocity command. Commands generated for individual wheel velocities, as required for our controllers, are decomposed into the requisite linear and angular velocity commands through a linear transformation and sent to the Segway base controllers. We wrote our software in Python and used OpenCV [13] and ROS (Robot Operating System) [14] both of which are open source.

B. The Camera and Light Ring

We used a Dragonfly2 camera with remote head from Point Grey Research with a 4 mm focal length micro-lens. We mounted the camera approximately 10 cm off the ground between the two RMPs (front and back), facing it directly downwards. The camera is capable of returning 1024x768 resolution, 8-bit color images at 30 fps. Using ROS we were able to grab frames and transmit them at full resolution at 30 fps. However, our practical loop rate for the control system was limited to 10 Hz due to feature extraction and tracking. Images from four different floors on which the odometry system worked are found in Figure 2

We designed a light ring using 16 1.3W Luxeon Star White Lambertian LEDs. We angled each LED downwards to the center of the luminous cone (see Figure 3) in the cameras field of view. We also mounted the light ring under the robot and centered it around the downward-facing camera (as seen in Figure 1). This allowed for illumination under the robot without causing primary reflections to become specular reflections on reflective surfaces such as wood flooring.

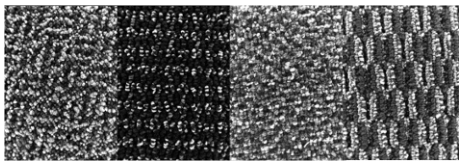

Fig. 2. Four examples of flooring that were imaged during the operation of our visual odometry system.

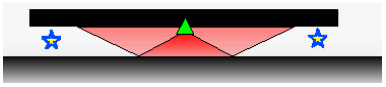

Fig. 3. Diagram of light ring showing showing how geometry helps avoid specular reflections.

C. Coordinate Frames

Coordinate frames play an important role in the derivation and explanation of the system, so we define the robot frame here. As shown in Figure 1, the robot's frame is located at the geometric center of the robot base. The x-direction points forward from the robot, the y-direction is to the robot's left. Following a right-handed coordinate system, the z-direction is up. We assume that the camera center is coincident with the origin of the robot's reference frame, a reasonable assumption due to the design of our mounting bracket. We also require that the xy-plane be parallel to the ground. To satisfy this requirement we calibrated the camera to account for small angular deflection errors in the mounting. To perform the calibration, we imaged a calibration grid with known physical dimensions that we placed on the floor, allowing us to extract known correspondences between points in physical-space and those in pixel-space. We then found a mapping (homography) between the original image and a rectified image plane that is parallel to the ground plane using OpenCV.

III. Measurement and Motion Model

A. Odometry Measurement Model

For our VO system we use the Harris corner descriptor [15] and the pyramidal Lucas-Kanade feature tracker [16] in OpenCV to extract features and then find putative correspondences from two consecutive raw camera images. The images are queried using a python process in order to grab them at frame rate. They are subsequently passed using ROS to another python process that does the feature tracking at approximately 10 Hz. If we assume that the robot only moves in the plane and that two consecutive images are coplanar, then the transformation between feature correspondences is described by a planar rotation (θ) about the z-axis and a translation ($t \in \mathbb{R}^2$). We then consider the $< X, Y >$ set of coordinates for 2D points known in both the previous $< X^p, Y^p >$ and next $< X^n, Y^n >$ frames. The goal is to reconstruct the coordinate frame transformation that describes the motion of the camera with respect to

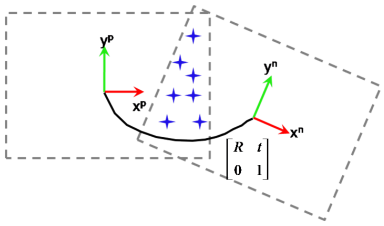

Fig. 4. Representative image of feature correspondences and their relationship from two consecutive frames.

the observed features. In homogeneous coordinates this is represented in Figure 4.

Linearizing the 2D homogeneous transformation equation under a small angle assumption yields:

$$\begin{bmatrix} 1 & \delta\theta & \delta x \\ -\delta\theta & 1 & \delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^p \\ y^p \\ 1 \end{bmatrix} = \begin{bmatrix} x^n \\ y^n \\ 1 \end{bmatrix} \tag{1}$$

Rearranging the rows of interest results in the following:

$$y^p \cdot \delta\theta + \delta x = x^n - x^p \tag{2}$$
$$x^p \cdot -\delta\theta + \delta y = y^n - y^p \tag{3}$$

By replicating the process for m pairs of points, we obtain:

$$\begin{bmatrix} 1 & 0 & y_0^p \\ 0 & 1 & -x_0^p \\ & \dots & \\ 1 & 0 & y_m^p \\ 0 & 1 & -x_m^p \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta\theta \end{bmatrix} = \begin{bmatrix} x_0^n - x_0^p \\ y_0^n - y_0^p \\ \dots \\ x_i^n - x_i^p \\ y_i^n - y_i^p \end{bmatrix} \tag{4}$$

Using the Moore-Penrose pseudo-inverse or overconstrained linear least squares solution to this problem minimizes the norm of the error. This result is valid assuming that all of the putative correspondences are valid inliers. In practice this will not likely be the case as the feature tracker appears to be very dependent on guesses ($< X^n, Y^n >$) provided to the feature tracking algorithm. To help reject gross outliers, we perform an initial estimate as described above. We can then calculate an approximate error for each feature correspondence by using the solution found to forward project the features in the previous image into the next and compare them with the actual features in order to compute an error in pixel units. We discard correspondences whose error is both greater than one pixel and greater than one standard deviation from the mean error, and then recompute a new odometry estimate using the remaining feature correspondences (presumably inliers).

We compute the current robot pose in an arbitrary but static global frame of reference by adding the latest VO estimate (in the robot's frame) to the previous step's global pose estimate (also expressed in the robot's frame). Being computed in pixel units, this pose is only known to a scale. Obtaining physically-meaningful units required an additional scaling factor that we determined during the calibration step described in Section II-C.

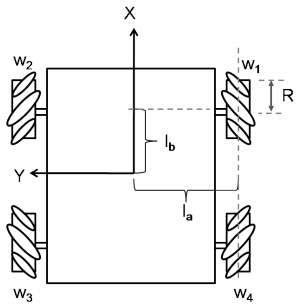

Fig. 5. Coordinate frame and labeling for Segway base with Mecanum wheels.

B. Kinematic Motion Model

The coordinate frame imposed on the mobile base shown in Figure 5 is also the robot coordinate frame from Figure 1 seen from above. Following the development of kinematic equations from [1], and according to the coordinate frame shown in Figure 5 we show that:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} 1 & 1 & l_{ab} \\ 1 & -1 & -l_{ab} \\ 1 & 1 & -l_{ab} \\ 1 & -1 & l_{ab} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{5}$$

Where w_i from Figure 5 is the i^{th} wheel of the base and ω_i is the associated angular velocity input from that wheel. R is the wheel radius which is 4 inches for our Mecanum wheels and l_{ab} is defined as $l_a + l_b$, the sum of the distances shown in Figure 5. The variables $\dot{x}, \dot{y}, \dot{\theta}$ are the linear velocities in the x and y directions and the angular velocity about the center of the robot. Equation 5 is referred to by Muir et al. [1] as the actuated inverse solution and is used to directly compute wheel velocities in order to attain a desired linear and angular velocity of the base. In commanding the base, we assume that the location of the robot when the odometry is initialized is the origin of a global coordinate frame. This is done without loss of generality since any frame of reference could be used at initialization or after a localization step. Velocity commands are then given in a global coordinate frame and afterwards transformed into the local robot frame as follows:

$$\begin{bmatrix} \dot{x_r} \\ \dot{y_r} \\ \dot{\theta_r} \end{bmatrix} = \begin{bmatrix} sin(\theta_g) & cos(\theta_g) & 0 \\ cos(\theta_g) & sin(\theta_g) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x_g} \\ \dot{y_g} \\ \dot{\theta_g} \end{bmatrix} \tag{6}$$

Where the subscripts g and r denote the coordinates in global and robot frame. The result is that we can command velocities in the global frame and decompose them at each discrete time step into the robot's local frame.

In order to later develop the Linear Quadratic Regulator (LQR) controller, we also need the prediction of the linear and angular velocities of the base given the angular velocities of the four wheels. In [1], Muir and Neuman call this the sensed forward solution because it is assumed that the wheel velocities are known. The relation established by Equation

5 is overconstrained and therefore the solution for the linear and angular velocities is a linear least squares approximation which is shown below:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \frac{R}{4\ l_{ab}} \begin{bmatrix} l_{ab} & l_{ab} & l_{ab} & l_{ab} \\ l_{ab} & -l_{ab} & l_{ab} & -l_{ab} \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \tag{7}$$

When slipping occurs, which is the mode of operation for Mecanum wheels, the equations no longer hold and are only an approximation. The major reason for our VO implementation is that the sensed forward solution is not adequate for controlling the base due to wheel slippage from uneven floors, uneven weight distribution on the base and various other reasons, as will be shown in the results section.

IV. Controller for Omnidirectional Base

A. PID Control

The first control scheme we implemented included three separate PID loops wrapped around the measured variables. This loop was the same for each controlled variable (x, y, θ) as shown in Figure 6. The reference input is a global pose (position or orientation). After being projected into the robot's frame the resultant control scalar from each of the three PID loops is concatenated to form $[\dot{x}\ \dot{y}\ \dot{\theta}]^T$. The angular velocities for the wheels corresponding to the desired velocity commands are then found through Equation 5. For both the PID controller and the LQR controller, the actuator or wheel velocities are limited to avoid VO failure due to blurred images or loss of putative correspondences. The feature tracker update rate currently limits the speed of the robot.

We tuned the proportional, integral and derivative gains for each loop individually using the common Ziegler-Nichols method [17]. After this initial tuning, we performed further iterations with all three loops active at the same time in order to reduce common oscillation, which is a known problem with this tuning method. There are two major drawbacks with the PID method. First, we treat each PID loop as being independent when, in fact, they are coupled. This can lead to undesired behavior or overshoot. Second, tuning the gains does not leverage information that may be provided by the known kinematics of the base.

B. LQR Control

We formulate our LQR controller using the kinematic equations previously shown for two reasons. First, with unequal weight distribution and uneven flooring the wheels slip in such a way that a useful dynamic model would be difficult if not impossible to obtain. Second, although the kinematic equations assume that velocities commanded are equal to velocities attainable in a finite time step (which is unrealistic in some scenarios), we have found it to work well for moderate accelerations. In order to develop the LQR controller, we discretized Equation 7. Assuming a simple first order Euler approximation the following become the discrete time state equations for the base:

$$\begin{aligned} \mathbf{x}(k+1) &= G\mathbf{x}(k) + H\mathbf{u}(k) \\ \mathbf{y}(k) &= C\mathbf{x}(k) + D\mathbf{u}(k) \end{aligned} \tag{8}$$

The state vector $\mathbf{x}(k)$ is:

$$\mathbf{x}(k) = [x(k)\ y(k)\ \theta(k)]^T \tag{9}$$

The matrices G, H, C and D are defined as:

$$\begin{aligned} G &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ H = \begin{bmatrix} t\ l_{ab} & t\ l_{ab} & t\ l_{ab} & t\ l_{ab} \\ t\ l_{ab} & -t\ l_{ab} & t\ l_{ab} & -t\ l_{ab} \\ t & -t & -t & t \end{bmatrix} \\ C &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ D = [0] \end{aligned} \tag{10}$$

with t being the change in time for each discrete step and the input $\mathbf{u}$ as:

$$\mathbf{u} = [\omega_1\ \omega_2\ \omega_3\ \omega_4]^T \tag{11}$$

Although we treated each state variable in the PID controller as a SISO (single input single output) system, this state space representation allows the use of MIMO state space theory to form a more theoretically rigorous and robust controller. Although the PID controller works in practice, it seems to have problems with oscillation that we hypothesize is due at least in part to the coupling between the rotation and translation since they are treated as being independent. Although it might be possible to tune the PID controller to obtain acceptable behavior, LQR control design allows us to model the MIMO system including the coupling of different states in a straightforward manner.

A controller formulated with the state space equations from Equation IV-B would be a regulator only and would drive the states to zero. We therefore introduce a reference input as seen in Figure 7. We can then define a new state vector with an integrator term included, (also in Figure 7):

$$\left[\hat{\mathbf{x}}(k)\right] = \begin{bmatrix} \mathbf{x}(k) - \mathbf{x}(N) \\ \sum_{k=0}^{N} (\mathbf{x}(k) - \mathbf{x}(N)) \end{bmatrix} = \begin{bmatrix} \mathbf{x_e}(k) \\ \sum_{k=0}^{N} \mathbf{x_e}(k) \end{bmatrix} \tag{12}$$

With the new state equations that introduce a reference input and integral action, we can formulate the performance index for a Linear Quadratic Regulator (LQR). We assume an infinite time horizon for the controller development which means that the LQR gain matrix, K will be constant and the solution is found by resolving the discrete algebraic Ricatti equation. Following common derivations for a discrete system such as found in [18] , we let the control input and performance index be defined as:

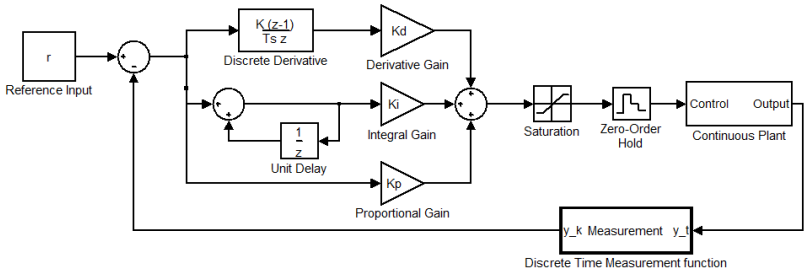

Fig. 6. Block diagram for a single loop of the PID controller.

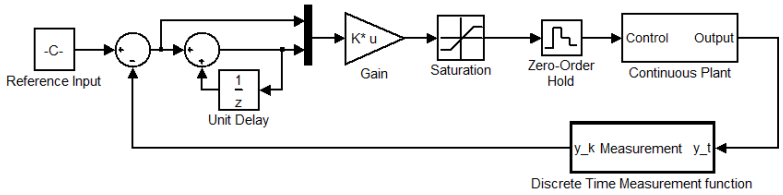

Fig. 7. Block diagram for LQR controller.

$$\begin{aligned} \hat{\mathbf{u}}(k) &= -K\hat{\mathbf{x}}(k) \\ J &= \tfrac{1}{2}\sum_{k=0}^{\infty}[\hat{\mathbf{x}}(k)^T Q\hat{\mathbf{x}}(k) + \rho\hat{\mathbf{u}}(k)^T R\hat{\mathbf{u}}(k)] \end{aligned} \quad (13)$$

We initially set Q equal to a $6x6$ identity matrix and R to a $4x4$ identity matrix. ρ defines the relative weighting between the two terms and was initially defined as $\rho = 1$. This weighting parameter was later modified after testing with the mobile base in the loop to find an acceptable response empirically. In addition, the relative weightings for the diagonal terms of Q corresponding to the states of x and y were weighted differently than for θ to account for the difference in units.

We implemented the LQR controller, shown graphically in Figure 7, in Python with a fixed time step t of $\frac{1}{10}$ of a second.

V. Validation of Control and Measurement System

We show the performance of the VO and the two controllers through a series of three tests. The results show the capabilities of the two controllers in conjunction with the performance of the visual odometry. We tuned the two controllers with hardware in the loop before any data was taken. In addition, both had anti-windup measures added which involved setting the integral control terms to zero if common anti-windup conditions were met. Doing this limited the amount of oscillation about the set points or waypoints due to the integral term. Our current implementation of visual odometry runs at 10 Hz. This means that the speed of the robot is limited to approximately 0.3 meters per second. Possible improvements to the system are addressed in Section V-D.

A. Small-Scale Motion and Waypoint Tracking

For each of the following results in this section, we recorded the odometry from both the VO system and from the sensed forward solution using wheel encoder values. In addition we used an overhead calibrated camera to show the performance of the odometry with respect to ground truth (see Figure 8). The resolution of the ground truth measurement is on the order of 1 cm and we believe it has some distortion problems at the edges of the field of view causing some error in the ground truth measurement. However it is not subject to the same effects of drift found in the dead reckoning estimates for the other odometry measures. We describe the test scenarios in each of the following sections, and they are represented graphically in Figure 8. The view in this figure is from the overhead camera. The red lines denote a translation in the direction of the arrow and the green curves denote a rotation of 90° in the indicated direction. For each test, we show the performance of the odometry with ground truth as well as showing the transient response for the PID and LQR controllers on each state. The results are in Figure 9 and the column divisions represent the three different tests and are labeled accordingly in the figure.

1) Waypoint Tracking: The first test consisted of four commands sent to the base in succession as waypoints. The commands formed a square with edge length of 1.1 meters and included a 90 degrees counter clockwise rotation for each of the first two edges and a negative 90 degrees rotation for the last two edges. This means that the robot should have ended in the pose where it began. The robot was given 20 seconds for each edge or waypoint since this was determined to be larger than the 95 percent rise time for both controllers even in the presence of actuator saturation.

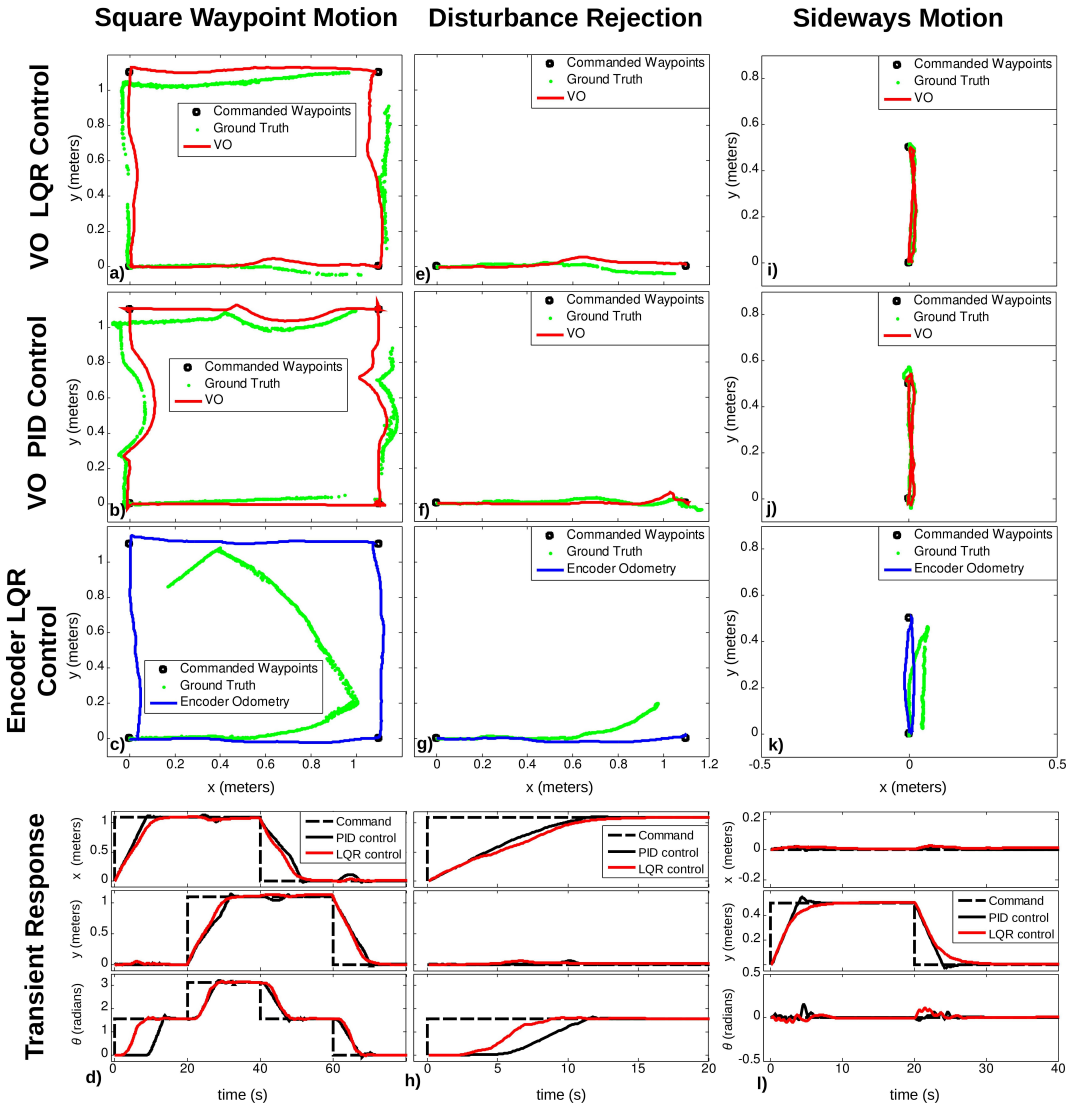

Fig. 9. These results are organized with the column defining the test completed (col 1 = Sideways Motion, col 2 = Square Waypoint Motion, col 3 = Disturbance Rejection). The first three rows also define the type of feedback and control used for these tests (row 1 = LQR control using VO, row 2 = PID control using VO, row 3 = LQR control using encoder odometry). Row 4 shows the transient response of the controllers for each state variable of interest across the three tests.

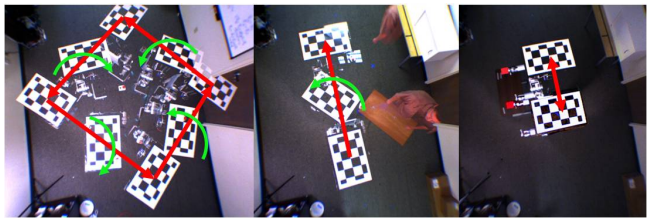

Fig. 8. View from overhead calibrated camera that was used to calculate ground truth for odometry. The three images represent the three test cases. Red lines denote translation and green arrows show a rotation of 90° in the indicated direction.

The size of the square defined by the four waypoints was restricted by the field of view of the overhead camera. We ran tests using visual odometry as the measurement system for both the LQR and PID controllers and the results in terms of estimated position are in Figure 9 (a) and (b).

We ran the test again with the LQR controller, this time using the encoder values from the Segway and the forward sensed solution from [1] to measure the states. The results for this test are shown in Figure 9(c). These first graphs show only the coupled results of the controller and the odometry system and no dependence on time is represented. For this reason we also included the transient response for the PID and LQR controller using the VO in Figure 9(d).

Having measured the visual odometry estimates, wheel-encoder odometry estimates, and ground truth for these scenarios, we were able to calculate the amount of closed-loop error for this short trajectory with 360° of total angular rotation. We defined the closed-loop error as the difference between the ground truth and the odometry estimate after the robot completed the square trajectory divided by the total distance estimated by the odometry method. We summarize the

TABLE I

CLOSED-LOOP ERROR BETWEEN GROUND TRUTH AND ODOMETRY ESTIMATE FOR A SINGLE RUN.

Odometry Type	Controller	% Error
VO	PID	0.31%
VO	LQR	0.52%
Encoder	PID	8.17%
Encoder	LQR	14.4%

results for a single trial of each test scenario in Table I. The closed-loop error for our visual odometry was under 0.52%, which clearly outperformed the wheel-encoder odometry.

2) Disturbance Rejection: In the next test we commanded only one waypoint but had a wooden board approximately 0.5 cm thick located in the wheel path to simulate a disturbance. For each test the board was located at the same spot to within approximately 1 mm of the other tests. This was not a rigorous or thorough way to measure disturbance rejection in the system. It did, however, give a qualitative idea for robustness of the combined controller and measurement feedback loop. As for the first test, the results are shown in Figure 9

3) Sideways Motion for Manipulation: In this test we tried to show the utility and feasibility of using the base with a mobile manipulator to increase redundancy in the kinematic chain. The robot end effector was extended over a table top. The base was then commanded to move 50 cm to the left (in the y-direction), then 50 cm back to where the manipulator started. Again the results are shown in Figure 9.

B. Discussion

The first important result for this system is that control using only the encoders was clearly deficient in terms of accuracy. After only a single way point from the first test (which included a 90 degree rotation and 1.1 meter translation), dead reckoning when controlling with the encoders began to drift from ground truth by about 20 percent of the total distance traveled to that point. However, control with either the PID or LQR controller using VO was able to complete the entire trajectory of waypoints with less than 1 percent error with respect to ground truth. Introduction of disturbances such as uneven flooring or change in location of the center of gravity would only worsen the problem of controlling from the encoder output as any extra slip would add to errors in the odometry estimate.

In the third test scenario, commanding straight line trajectories in the y or sideways direction and controlling with encoder values caused the robot to move along an arc rather than a straight line. We hypothesize that this is due to the fact that the center of mass is shifted towards the back of the base causing greater friction contact forces on the back wheels. This same trend does not appear on the trajectories where we controlled using VO.

A number of factors contribute to the viability of our VO system for improving odometry on any mobile system with indoor applications. Clearly cameras are ubiquitous and cheap. In addition the LED light ring is a simple and reproducible design. It is feasible that this type of odometry could be a cost-effective and robust alternative to more expensive systems such as high resolution wheel encoders or odometry based on laser range scanners. Ease of use is also an important factor as the only calibration necessary to obtain reasonable results is finding the scaling factor and homography to rectify the image with respect to the ground plane. Another factor is that failure modes that are common for other visual odometry systems are less of a concern for our system. Poor lighting and crowded rooms do not change the VO performance since the robot must be able to physically occupy the area from which it measures and is able to control the lighting in that space.

C. Comparing LQR and PID Controllers

Very little can be said in comparing the LQR and PID controllers. One obvious observation is that the LQR controller appears to produce less oscillation and overshoot than the PID controller. One reason for this might be because the Ziegler-Nichols method is known to exhibit quarter wave decay. We tried to do additional tuning manually to minimize the effects of oscillation, but because the output was coupled it was difficult to tune in a satisfactory manner. For this reason, the LQR controller was preferable because it was simpler to tune the gains in a structured way.

D. Improvements on Current VO Implementation

There are many ways that the VO system could be improved. One way is that feature tracking at 30 fps has been shown for 1000 features on a 1024x768 resolution image [19]. Speeding up the loop rate in this way would improve the control and increase the maximum speed of the robot. This in addition to more robust methods (RANSAC or Kalman filters) for the odometry estimate would produce better quality and more robust odometry.

E. Limitations of Current Platform

Although our controller formulation and visual odometry system allowed us to overcome some of the obstacles for useful applications of Mecanum bases, other obstacles remain. After the initial disturbance tests shown in Figure 9 we determined that further tests would be useful to prove the robustness of the controllers and VO system. We placed three disturbances that would commonly be found in a household (a throw rug, a threshold and an extension cord) in the path of the base. Our original intention was to have the base move in a sideways trajectory over these objects. However, we were unable to even teleoperate the base sideways across the threshold as the Mecanum base became inoperable due to high-centering. In addition, the extension cord was quickly pulled up into the wheels because of spacing in the wheel design. Finally, we also noticed that as the wheels turn in opposite directions to move the base sideways, they are likely to pull up throw rugs and damage our VO system or at least detach it. We did not include quantitative results for these tests. However, it is important to note that these types of

mechanical limitations would be important in determining appropriate applications of the Mecanum base.

While this paper does not perform a thorough examination of these limitations, further investigation is certainly warranted. Further development should include more rigorous testing of disturbance inputs using difficult terrain such as ramps that are common in wheelchair accessible environments or throw rugs that cause uneven ground or more importantly can be pulled up between the two Mecanum wheels. Our experience suggests that Mecanum bases and the techniques discussed in this paper are most applicable to industrial or office settings with fairly flat flooring that is uncluttered.

VI. Conclusions

In this paper we have presented a visual odometry system that is accurate enough to give 0.52% closed-loop error over small trajectories. Using this visual odometry we have shown the development and performance of PID and LQR controllers for a Mecanum omnidirectional base. This type of control has already allowed us to successfully perform mobile manipulation tasks. Finally, because of the design of our system, it is possible for the odometry and controllers to function in environments with moving people and varying or low ambient light.

VII. Acknowledgement

We gratefully acknowledge support from the US National Science Foundation (NSF) grants CBET-0932592 and IIS-0705130, and support from Willow Garage.

References

[1] P. F. Muir and C. P. Neuman, "Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot," pp. 25–31, 1990.

[2] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive gps," in *International Conference on Pattern Recognition (ICPR)*. Citeseer, 2006.

[3] K. Konolige, M. Agrawal, and J. Sola, "Large scale visual odometry for rough terrain," in *Proc. International Symposium on Robotics Research*. Citeseer, 2007.

[4] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa, "A robust visual odometry and precipice detection system using consumer-grade monocular vision," in *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, vol. 3. Citeseer, 2005, p. 3421.

[5] D. Sekimori and F. Miyazaki, "Self-localization for indoor mobile robots based on optical mouse sensor values and simple global camera information," in *2005 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2005, pp. 605–610.

[6] J. Hu, Y. Chang, and Y. Hsu, "Calibration and on-line data selection of multiple optical flow sensors for odometry applications," *Sensors & Actuators: A. Physical*, vol. 149, no. 1, pp. 74–80, 2009.

[7] M. Dille, B. Grocholsky, and S. Singh, "Outdoor Downward-facing Optical Flow Odometry with Commodity Sensors," 2009.

[8] X. Song, L. Seneviratne, K. Althoefer, Z. Song, and Y. Zweiri, "Visual odometry for velocity estimation of UGVs," in *International Conference on Mechatronics and Automation*, 2007.

[9] J. Cooney, W. Xu, and G. Bright, "Visual dead-reckoning for motion control of a Mecanum-wheeled mobile robot," *Mechatronics*, vol. 14, no. 6, pp. 623–637, 2004.

[10] H. Wang, K. Yuan, W. Zou, and Q. Zhou, "Visual odometry based on locally planar ground assumption," in *2005 IEEE International Conference on Information Acquisition*, p. 6.

[11] A. Kelly, "Mobile robot localization from large-scale appearance mosaics," *The International Journal of Robotics Research*, vol. 19, no. 11, p. 1104, 2000.

[12] A. Jain and C. C. Kemp, "Pulling Open Doors and Drawers: Coordinating an Omni-directional Base and a Compliant Arm with Equilibrium Point Control," in *ICRA*, 2010.

[13] G. Bradski, V. Pisarevsky, and J. Bouguet, "Open source computer vision library," *Intel Corporation*, 2001.

[14] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, R. W. Eric Berger, and A. Ng, "ROS: an open-source Robot Operating System," in *Open-Source Software workshop of (ICRA)*, 2009.

[15] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15. Manchester, UK, 1988, p. 50.

[16] J. Bouget, "Pyramid Implementation of the Lucas Kanade Feature Tracker," Technical report, Tech. Rep. included in the OpenCV library, Tech. Rep., 2002, http://www. intel. com/research/mrl/research/ovencv. 3, Tech. Rep.

[17] J. Ziegler and N. Nichols, "Optimum settings for automatic controllers," *Journal of dynamic systems, measurement, and control*, vol. 115, p. 220, 1993.

[18] K. Ogata, *Discrete-time control systems*. Prentice-Hall Englewood Cliffs, NJ, 1987.

[19] S. Sinha, J. Frahm, M. Pollefeys, and Y. Genc, "GPU-based video feature tracking and matching," in *EDGE, Workshop on Edge Computing Using New Commodity Architectures*, vol. 278. Citeseer, 2006.