Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Planning and Control of Mobile Robots in Image Space from Overhead Cameras*

Rahul S. Rao, Vijay Kumar, Camillo J. Taylor
General Robotics, Automation, Sensing and Perception (GRASP) Laboratory
University of Pennsylvania, Philadelphia, Pennsylvania 19104, U.S.A
{*rahulrao, kumar, cjtaylor*}*@grasp.cis.upenn.edu*

***Abstract*— In this work, we present a framework for the development of a planar mobile robot controller based on image plane feedback. We show that the design of such a motion controller can be accomplished in the image plane by making use of *a subset of* the parameters that relate the image plane to the ground plane, while still leveraging the simplifications offered by modeling the system as a differentially flat system. Our method relies on a waypoint-based trajectory generator, with all the waypoints specified in the image, as seen by an overhead observer. We present some results from simulation as well as from experiments that validate the ideas presented in this work and discuss some ideas for future work.**

***Index Terms*— Visual servoing, differential flatness, trajectory generation and control.**

I. Introduction and Related Work

There has been a great interest of late in the development of vision as a means of sensing and feedback for autonomous robotic systems. The availability of enhanced computing power, coupled with the falling costs of computers as well as sensors, has been greatly responsible for this significant development. As we all know, vision is a key tool that we use in our daily lives. Thus, incorporating robotic systems with vision and the ability to make real-time decisions on the basis of vision in addition to other sensors will greatly enhance their ability to perform more complex tasks. Beginning with the development of strategies for the real time control of robot arms to reach desired configurations with the aid of attached cameras, cameras today are used extensively for the real time control of not only robot arms but also mobile robots, groups of mobile robots as well as in cars, such as for lane departure warning systems as well as aids for parallel parking technologies in cars.

Visual servoing is the fusion of several areas of research. These include high speed image processing, kinematics, dynamics, control theory and real time computing. Active vision and structure from motion are also closely related areas of research. Applications that have been proposed and implemented in the realm of visual servoing include grasping objects on conveyor belts, juggling [13], part mating [14], teleoperation, missile tracking, fruit picking [7], robotic ping-pong [1], car steering [9] and even aircraft landing ([15], [6]).

*This work was supported by ARO MURI Grant DAAD19-02-010383 and NSF Grant IIS00-83240

In our earlier work in this area we have presented a framework for image based control of a wheeled ground vehicle from an overhead camera. We have presented the background and the problem formulation in the image space [10] with some preliminary results in simulation. We also presented some insights into and the significance of the relationship between the robot's ground plane velocity and the velocity of its projection in the image and how control tasks could be reformulated in the realm of this limited parameter space [11]. Results from simulation and experimental validation of our ideas as well as their applications in some control tasks were also presented. We have also presented results from the implementation of an image based point to point motion controller in real time to the case of a moving overhead camera with unstructured motion [12].

In this work, we will be highlighting the significance of the problem formulation in terms of the limited parameters of the 2-D homography (rather than the *entire* homography) from the perspective of trajectory generation and differential flatness based motion control *along* this trajectory. Section II provides an overview of the relationship between visual servoing and differential flatness. Section III describes how the controller can be designed in the image space utilizing the simplifications offered by modeling the system as a differentially flat one. Section IV describes the trajectory generation scheme that we use in tandem with the controller design discussed earlier in order to define a desired trajectory for the robot to follow. In Sections V and VI, we present some results from simulation and real time experiments that are all built upon the ideas presented in prior sections. Finally, we discuss some ideas for future work in Section VII.

II. Background

A. Projective Geometry

Let $\mathbf{w} \equiv (x, y, 1)^T$ denote the homogeneous coordinates of a point on the ground plane and $\mathbf{c} = (u, v, 1)^T$ denote the coordinates of the projection of $\mathbf{w}$ in the image. It is easy to show that $\mathbf{w}$ and $\mathbf{c}$ are related by a projective transformation $\mathbf{G}$. This can be expressed as

$$\mathbf{c} \propto \mathbf{G}\mathbf{w}, \ \mathbf{G} \in GL(3) \quad (1)$$

$$\Rightarrow \mathbf{w} \propto \mathbf{H}\mathbf{c} \quad (2)$$

where $\mathbf{H} = \mathbf{G}^{-1}$.

For clarity, let the matrices $\mathbf{G}$ and $\mathbf{H}$ be represented in terms of their columns as $\mathbf{G} = \begin{pmatrix} G^1 & G^2 & G^3 \end{pmatrix}$ and $\mathbf{H} = \begin{pmatrix} H^1 & H^2 & H^3 \end{pmatrix}$ respectively. Similarly, let the rows of $\mathbf{G}$ and $\mathbf{H}$ be represented by $\begin{pmatrix} G_1 & G_2 & G_3 \end{pmatrix}$ and $\begin{pmatrix} H_1 & H_2 & H_3 \end{pmatrix}$ respectively. Note that superscripts and subscripts are used to distinguish between matrix columns and rows.

We have shown in our earlier work [10] that the relationship between ground velocities, $(\dot{x}, \dot{y})$, and image plane velocities, $(\dot{u}, \dot{v})$, can be written as

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = (H_3 \cdot \mathbf{c}) \begin{pmatrix} 1 & 0 & -u \\ 0 & 1 & -v \end{pmatrix} \begin{pmatrix} G^1 & G^2 \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \quad (3)$$

where

$$H_3 = \frac{G^1 \times G^2}{(G^3) \cdot (G^1 \times G^2)} = \frac{G^1 \times G^2}{det(\mathbf{G})} \quad (4)$$

Since $\mathbf{G}$ represents a projective transformation, it's scale is immaterial which means that we can, without loss of generality, restrict $\mathbf{G}$ to be such that $det(\mathbf{G}) = 1$. As described in our previous work [11], we use this condition and the property that the right hand side of equation (3) must be perpendicular to the vector $\begin{pmatrix} -\dot{v} & \dot{u} \end{pmatrix}$ to obtain the relationship

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \alpha^3((\hat{G}^1 \times \hat{G}^2) \cdot \mathbf{c}) \begin{pmatrix} 1 & 0 & -u \\ 0 & 1 & -v \end{pmatrix} \begin{pmatrix} \hat{G}^1 & \hat{G}^2 \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \quad (5)$$

where $\begin{pmatrix} G^1 & G^2 \end{pmatrix} = \alpha \begin{pmatrix} \hat{G}^1 & \hat{G}^2 \end{pmatrix}$. Thus, we get

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \mathbf{Q}_{2\times 2} \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix} \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} \quad (6)$$

B. Differential Flatness

The basic idea of a differentially flat system was introduced by Fliess et. al [5]. A differentially flat system is suited for trajectory generation tasks [16] including real time feasible trajectory generation in the presence of inequality constraints [4]. The reader is referred to ([5], [16]) for the formal mathematical definition of a differentially flat system.

Murray et. al [8] provide many examples of mechanical systems that are differentially flat. As shown by Nieuwstadt and Murray [16], it is relatively simple to generate trajectories for differentially flat systems. This can be done even when there are inequality constraints on the state [4].

With reference to differential flatness, we have shown in earlier work [10] that the image based system is differentially flat with the *image coordinates* of the center of the rear axle, (u, v), of the unicycle being the differentially flat outputs. This translates to the fact that the states of the system, (x, y, ϕ) and the system inputs, (ν, ω), can be explicitly represented as functions of the flat outputs and their higher derivatives. Details are provided in [10], but it can be summarized as

$$[x, y, \phi, \nu, \omega]^T = [(f_1(u, v, \dot{u}, \dot{v}, \ddot{u}, \ddot{v}) \dots f_5(u, v, \dot{u}, \dot{v}, \ddot{u}, \ddot{v})]^T \quad (7)$$

III. Controller Design

There exists a methodology to design controllers for differentially flat systems [16]. We take a slightly different approach toward design in the image space. The key is that we do the path generation, controller design and the system feedback all in the flat space. In contrast to previous work [16], our controller design requires feedback from the image plane, which is also the flat space. Further, we compute the real world inputs $(\dot{\nu}, \omega)$ by first computing the inputs in the flat space and then using a nonlinear transformation to obtain $(\dot{\nu}, \omega)$. The entire procedure representing the trajectory generator, the controller and the feedback loop is illustrated in Figure 1 and the details of the controller design are presented next.

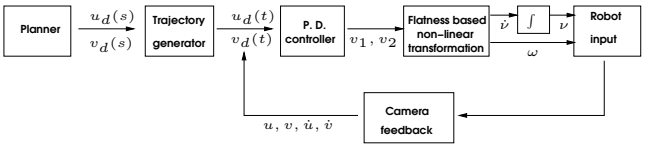

Fig. 1. The feedback loop for flatness based control of a robot along a desired trajectory.

We first observe that Equation (5) presents a relationship between the image velocities and the ground velocities of the robot. However, when combined with the kinematic model of a unicycle [10], it is observed that the input ω does not appear on the right hand side of Equation (5).

Thus, we need to differentiate Equation (5) once more to relate the outputs of our system, (u, v) with the inputs, (ν, ω), i.e. our system is of relative degree 2.

Differentiating Equation (5) with respect to time once again so that the second input ω also appears in the dynamics of the system, we obtain

$$\begin{pmatrix} \ddot{u} \\ \ddot{v} \end{pmatrix} = \mathbf{T} \begin{pmatrix} \dot{\nu} \\ \omega \end{pmatrix} + \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \equiv \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \quad (8)$$

where v_1 and v_2 are system inputs in the flat space. These are *not* the inputs to the actual, physical robot. These physical inputs, namely the forward and turning speeds (ν and ω respectively) are then obtained using nonlinear transformations that are a direct consequence of the system being differentially flat.

Once the flat space inputs, (v_1, v_2), are computed, they can be transformed into real world inputs, namely forward and angular speeds (ν and ω, respectively). Equation (8) translates to

$$\begin{aligned} \begin{pmatrix} \dot{\nu} \\ \omega \end{pmatrix} &= \mathbf{T}^{-1} \begin{pmatrix} v_1 - \alpha_1 \\ v_2 - \alpha_2 \end{pmatrix} = \mathbf{P} \begin{pmatrix} v_1 - \alpha_1 \\ v_2 - \alpha_2 \end{pmatrix} \\ &= \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} \begin{pmatrix} v_1 - \alpha_1 \\ v_2 - \alpha_2 \end{pmatrix} \end{aligned} \quad (9)$$

Thus, the explicit representations for the real world inputs are

$$\dot{\nu} = P_{11}(v_1 - \alpha_1) + P_{12}(v_2 - \alpha_2) \quad (10)$$

$$\omega = P_{21}(v_1 - \alpha_1) + P_{22}(v_2 - \alpha_2) \quad (11)$$

We note here that in order to provide a translational velocity input (ν) to the robot in the real world, $\dot{\nu}$ will have to be integrated over time, as shown in Figure (1).

In summary, as a consequence of the system being differentially flat in the image space, the real world inputs ν and ω can be represented by algebraic transformations that are functions of image coordinates, image velocities and robot orientation.

It is natural to ask how one calculates the inputs in the flat space, v_1 and v_2. To compute the flat space inputs that will control the robot about the desired trajectory (call it $(u_d(t), v_d(t))$), we can make use of the linear representation above (Equation (8)).

So, if Equation (8) is rewritten in state space form, it will appear as

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= v_1 \\ \dot{z}_3 &= z_4 \\ \dot{z}_4 &= v_2 \end{aligned} \tag{12}$$

where $(z_1, z_2, z_3, z_4) \equiv (u, \dot{u}, v, \dot{v})$. A proportional derivative (PD) controller can be chosen to be of the following form that incorporates feedforward as well as feedback terms,

$$v_1 = \ddot{z}_{1d} + k_1(\dot{z}_{1d} - \dot{z}_1) + k_0(z_{1d} - z_1) \tag{13}$$

$$v_2 = \ddot{z}_{3d} + k_3(\dot{z}_{3d} - \dot{z}_3) + k_2(z_{3d} - z_3) \tag{14}$$

$(u_d(t), v_d(t)) \equiv (z_{1d}(t), z_{3d}(t))$ is the desired trajectory that is obtained by a trajectory generation scheme. The feedforward terms keep the robot on the desired trajectory while the feedback terms get it back on track if it goes off the trajectory due to modeling errors, system noise etc.

Having transformed the problem from its original nonlinear structure to a linear framework, the task of designing a linear state observer in the flat space can be accomplished using linear systems theory. The rate of convergence of the estimator can be controlled so that the estimated state approaches the actual state at a desired rate by a proper choice of the observer gain matrix [3].

We now discuss a scheme to generate desired trajectories in the $2D$ image space, using parametrization in terms of the arc length along the curve, s, as a parameter.

IV. Path Generation in the Image Space

In order to follow a certain path in the $2D$ image space, one can define intermediate waypoints and then generate a curve that passes through these waypoints while maintaining certain continuity conditions at these waypoints (velocity continuity in our case). Once these waypoints are defined and the paths generated, control schemes can be used following the methods outlined in the previous section to guide the robot along the desired path. A method that parametrizes the desired path in terms of the arc length along the path, s, is now discussed in greater detail.

If each segment is assumed to be quadratic and of the form $u(s) = a_2 s^2 + a_1 s + a_0$, it is required that the 3 unknown coefficients a_2, a_1, a_0 be determined using continuity conditions at the waypoints. In order to determine the $3n$ coefficients, we will need $3n$ independent equations that can be solved simultaneously. This set of equations can be obtained by writing the expressions for the n segments, each of which passes through 2 points. $(n-1)$ more equations can be obtained by imposing continuity conditions at the intermediate points. The final equation is obtained by assuming a linear final segment of the curve.

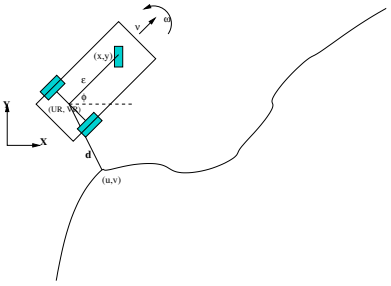

Fig. 2. A situation in which the robot is off the desired path, highlighting the closest distance d between the robot and the desired path.

Once the quadratic splines have been determined, the problem of determining the desired coordinate on the path, the desired velocity at a certain time instant and the desired acceleration at that instant can be formulated as a minimization problem as follows (Figure 2):

Given a certain location of the robot, (u_R, v_R), determine the location (u, v) on the piecewise quadratic path that is closest to the robot's current location, (u_R, v_R), i.e.

$$\begin{aligned} \text{Min} \quad & (u - u_R)^2 + (v - v_R)^2 \\ \text{such that} \quad & u = a_2^i s^2 + a_1^i s + a_0^i \\ \text{and} \quad & v = b_2^i s^2 + b_1^i s + b_0^i, \\ & i = 1, \ldots, n \end{aligned} \tag{15}$$

The goal of the optimization problem is to minimize the distance between the robot and the desired trajectory in the direction perpendicular to the trajectory as well as along the trajectory. The representation of the arc length as a function of time (such as $s(t) = \kappa t$) ensures that once the robot approaches the desired trajectory, it will continue to move along the desired trajectory with a certain speed.

This optimization problem, when expanded, results in having to find the roots of a third order polynomial, for which closed form solutions exist. Thus, a value of s will be the solution to this optimization problem, which will translate into a value for $(u(s), v(s))$.

Once $s(t)$ is obtained, $(\dot{u}_d(t), \dot{v}_d(t))$ can be obtained using the chain rule. Following that, Equations (13) and (14) can then be applied to obtain the values of the inputs that need be applied in the flat space domain to get the robot to follow the desired path.

We present now a quick summary of the minimization scheme to determine the point (u, v) on the curve that is closest to (u_R, v_R) at any time t. The objective function,

ϕ_o is defined as

$$\phi_o = (u - u_R)^2 + (v - v_R)^2$$

and the point (u, v) is constrained to lie on a segment of the curve. This translates to

$$\begin{aligned} u(s) &= a_2^i s^2 + a_1^i s + a_0^i \\ v(s) &= b_2^i s^2 + b_1^i s + b_0^i \end{aligned}$$

for some unknown $i \in [1, \ldots, n]$. The optimization problem will have to be solved for each segment of the spline as we do not know beforehand where the optimal solution to the problem lies, on the spline. The generic procedure described below can be applied to each segment. Once multiple solutions are obtained, infeasible solutions can be discarded if they are found to violate any feasiblity conditions. For instance, one might obtain $s < 0$ as a solution belonging to a segment of the curve. This solution is not physically possible as s, being the arc length along the curve, has to be greater than zero. Thus, the objective function is modified as

$$\begin{aligned} \phi_o &= (u - u_R)^2 + (v - v_R)^2 + \lambda_1(u - (a_2 s^2 + a_1 s + a_0)) \\ &\quad + \lambda_2(v - (b_2 s^2 + b_1 s + b_0)) \end{aligned} \tag{16}$$

The necessary conditions for the existence of an optimal solution are

$$\frac{\partial \phi_o}{\partial u} = 0;\ \frac{\partial \phi_o}{\partial v} = 0;\ \frac{\partial \phi_o}{\partial \lambda_1} = 0;\ \frac{\partial \phi_o}{\partial \lambda_2} = 0;\ \frac{\partial \phi_o}{\partial s} = 0$$

Evaluating these expressions, using the expressions for $u(s)$ and $v(s)$ now, and then simplifying, we obtain

$$\begin{aligned} &2s^3 + 3\frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} s^2 + \frac{2a_2(a_0 - u_R) + a_1^2 + 2b_2(b_0 - v_R) + b_1^2}{a_2^2 + b_2^2} s \\ &+ \frac{a_1(a_0 - u_R) + b_1(b_0 - v_R)}{a_2^2 + b_2^2} = 0 \end{aligned} \tag{17}$$

This is a cubic equation in s, and a closed form solution can be found [17]. However, there may be multiple solutions for s, translating into multiple solutions for (u, v). So, some of the solutions will be have to be discarded if they are not optimal or if $s < 0$ (since s is the arc length, it does not make sense for s to be negative).

Once a solution s is found, it will be found to *belong* to a certain segment of the piecewise quadratic spline. Thus, $u(s)$ and $v(s)$ can now be found, and so can their slopes and second derivatives. Thus, $u_d'(s)$ and $v_d'(s)$ can be found.

V. Path Following- Simulation Results

Section IV outlined the path generation in the image plane and Section III described the control of a robot along such a trajectory. We restrict ourselves to the use of second order splines as we intend to use kinematic vehicle models and velocity inputs to control the robot. We have implemented these ideas in simulation as well as in real time and we present some of the results that have been obtained in simulation and experiments.

We carried out simulations that generated paths made up of piecewise quadratic splines as described in Section IV and simulated controllers to track robots along those paths. An illustration of this path following method is shown in Figure 4. It must be noted that these waypoints

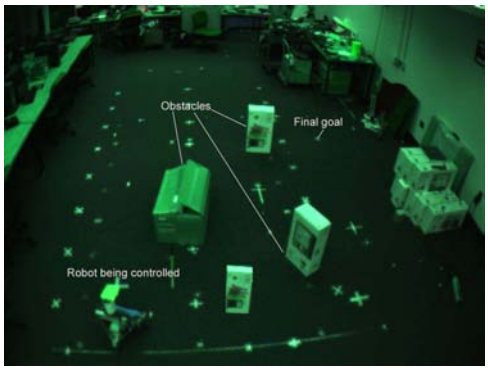

Fig. 3. A view from an overhead camera showing the robot, the obstacles and the desired goal. A shortest path algorithm can potentially be computed by defining appropriate waypoints *in the image*.

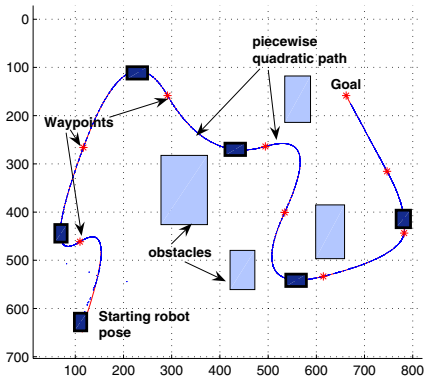

Fig. 4. A robot following a piecewise quadratic path generated in simulation using waypoints. Note that the robot starts at a point *off* the path and then converges to it, employing the optimization scheme described in Section IV.

are defined in the *image space* by just clicking on points in the image. Thus, a user can potentially get a shortest path to the goal by clicking on appropriate waypoints in the image. The path followed by the controlled robot in simulation corresponding to the overhead view depicted in Figure 3 is shown in Figure 4. The variation in ν and ω, the forward and turning speeds of the robot respectively, are also illustrated in Figure 5.

An alternate way to define the trajectory is to use an explicitly time parametrized trajectory rather than parametrizing it in terms of the arc length along the curve. Though hard to implement in practice due to factors such as system delays and system noise, such parameterizations validate the idea of using flatness based techniques for image based trajectory control.

Though the desired time parametrized trajectory is specified (such as a circle, ellipse or a path created to resemble a real life situation, such as in Figure 6), the starting position and orientation of the robot in the real world is chosen arbitrarily, the only constraint being that it should lie within the camera's field of view. The full state observer described briefly in Section III is used to establish the state vector.

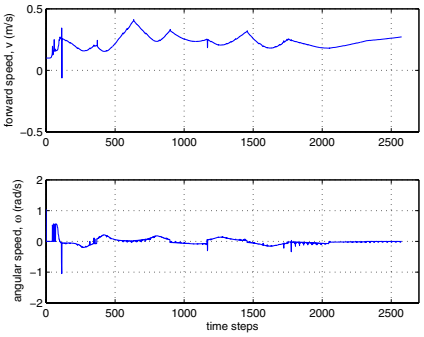

Fig. 5. Variation of forward and turning speeds of the robot as it follows the piecewise quadratic path shown in Figure 4.

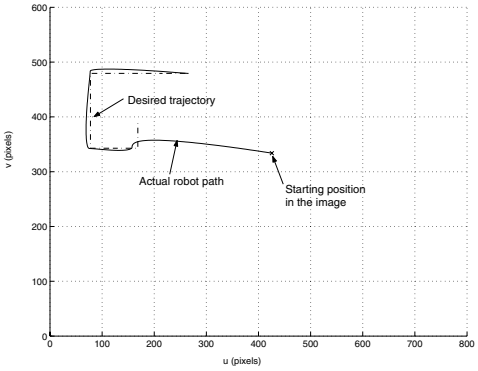

Fig. 6. The robot path as seen in the image (solid) and the desired image plane trajectory

Figure 6 shows the path traced by the robot, as seen in the image, under the influence of the controller designed using the procedure described in Section III. The starting position of the robot is indicated by a cross-hair and it can be seen that the robot does not start on the desired trajectory but converges to the desired trajectory.

VI. Path Following- Experimental Results

We also conducted experiments in real time to validate this framework of motion based calibration of the camera system coupled with the use of flatness based techniques for trajectory control of a mobile robots. Experiments have been conducted in real time using a distributed framework that is being developed independent of this work at the University of Pennsylvania. Further details of this framework for multi robot perception and control can be found in [2]. Experiments were conducted with the ER-1 mobile robot from Evolution Robotics and an overhead Dragonfly firewire camera with a Sony ICX204 sensor. The ER-1 is a three- wheeled, Hilare-like robot.

The experimental procedure to control the robot included the tasks of calibrating the system using motion correspondences, generating a trajectory as described in Section IV by defining waypoints in the image plane, computing the controller inputs as described in Section III and utilizing the feedback obtained to guide the vehicle along the desired

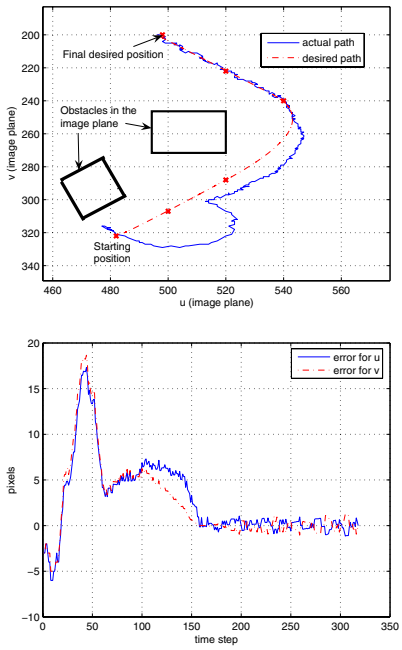

Fig. 7. A sample path taken by the robot to reach desired goals in the presence of obstacles (top) and the error between the actual and desired robot position in the image plane.

path. This procedure is illustrated in Figure 1.

Sample experimental runs are illustrated in Figures 7 and 8. In both cases, the waypoints are indicated by crosses (i.e. $\times$). The path followed by the robot enroute to its desired final destination is indicated by a solid line, while the desired trajectory generated in real time is indicated by a dashed line. We now make some observations from our experiments about the performance of our controller.

Any system operating in real time will be subject to some errors and uncertainties that can be reduced but not totally eliminated. There will be errors due to inherent flaws in the camera (pixel offsets, skew etc.). Image and feedback noise are a potential source of error. Real time calibration procedures are not ideal and can introduce errors. In addition, there will be errors in the implementation of the controller in real time due to time delays, input saturation etc.

We observe that the flatness based controller designed and implemented in the image space in real time does the task that it is required to do. Like any real time system, it is limited by the factors mentioned above. The robot tries to follow the desired trajectory (which is continuously updated according to the procedure outlined in Section IV) after the system is calibrated. While the robot is in motion, the feedback is obtained from the image, used to update the desired trajectory and to compute the inputs to be fed to the robot.

As mentioned in previous work [12], the same procedure could be implemented on an overhead camera mounted on a moving aerial observer. A simple image stabilization scheme that makes use of at least 4 fixed points that can

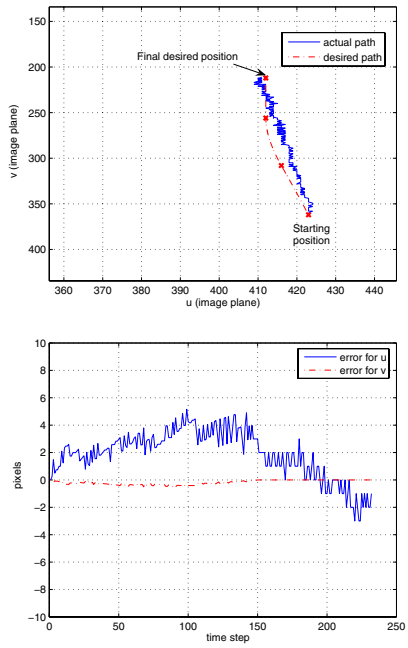

Fig. 8. A case of a simple straight path defined by image waypoints being followed by the robot (top) and a measure of the error between actual and desired image positions as measured in the image plane.

be tracked and identified over time has been implemented in real time.

VII. Conclusions and Future Work

In this work, we have presented a systematic procedure to combine the advantages offered by the representation of a system in a differentially flat manner with its representation in the image plane rather than the real world. We have presented a trajectory generator that utilizes the simplifications of the differentially flat representation to convert flat space inputs to real world inputs via simple algebraic transformations. The trajectory generator is novel by virtue of the fact that it is generated entirely in the image plane and the waypoints that are required to generate the trajectory are defined by just simple mouse clicks in the image plane. Further, the goal of the trajectory generator is such that the difference between the robot's location at any time and it's desired location at that time is minimized in the direction perpendicular to the trajectory as well as tangential to the trajectory, so that the robot continues to move along the trajectory once it gets to the desired position. We have presented results from simulations that incorporate these ideas as well as some experimental results that demonstrate real time trajectory tracking using the flatness based controller designed in the image space. In contrast to flatness based controllers that have been designed in the past, our method uses feedback from the image, i.e. the flat space to obtain flat space inputs first and then uses nonlinear transformations (obtained by using differential flatness) to obtain the real world robot inputs (forward and angular velocities).

A potential application of the ideas presented in this work relates to the control of groups of robots. None of the control strategies for control of formations of robots have yet leveraged the advantages of air-ground coordination. The global perspective than an overhead camera or a network of overhead cameras provides can potentially be combined with ideas from line-of-sight vision from cameras mounted on robots.

From a systems perspective the development of a system consisting of a network of cameras that communicate with each other while keeping the robots and the robot workspace within their combined field of view would enable the camera vision system to broaden its field of view and thereby allow for a greater range of motion of the ground robot. As one camera loses sight of the robot, the information that it possesses at that instant can be communicated to the next camera that is able to see it, thereby allowing for a smooth exchange of feedback information within the camera network.

References

[1] R. Andersson. *Real Time Expert System to Control a Robot Ping-Pong Player*. Ph.D. thesis, University of Pennsylvania, 1987.

[2] L. Chaimowicz, A. Cowley, V. Sabella, and C. Taylor. ROCI: A distributed framework for multi-robot perception and control. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, October 2003.

[3] C.-T. Chen. *Linear System Theory and Design*. Oxford University Press, 1999.

[4] N. Faiz, S. K. Agrawal, and R. M. Murray. Trajectory planning of differentially flat systems with dynamics and inequalities. *AIAA Journal of Guidance, Navigation and Control*, 24(2):219–227, 2001.

[5] M. Fliess, J. Levine, and P. Rouchon. Flatness and defect of nonlinear systems: Introductory theory and examples. *International Journal of Control*, 61(6):1327–61, 1995.

[6] R. Frezza and C. Altafini. Autonomous landing by computer vision: an application of path following on se(3). In *Proceedings of the 39th IEEE Conference on Decision and Control*. IEEE, 2000.

[7] R. Harrell, D. Slaughter, and P. Adsit. A fruit-tracking system for robotic harvesting. *Machine Vision and Applications*, 2:69–80, 1989.

[8] R. M. Murray, M. Rathinam, and W. Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. In *Proc. ASME Int. Mech. Eng. Congress and Expo*, San Francisco, November 1995.

[9] D. Pomerleau and T. Jochem. Rapidly adapting machine vision for automated vehicle steering. *IEEE Expert*, 11:19–27, April, 1996.

[10] R. S. Rao, V. Kumar, and C. J. Taylor. Visual servoing of a UGV from a UAV using differential flatness. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, October 2003.

[11] R. S. Rao, C. J. Taylor, and V. Kumar. Calibrating an air-ground control system from motion correspondences. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, D. C., June 2004.

[12] R. S. Rao, C. J. Taylor, and V. Kumar. Experiments in robot control from uncalibrated overhead imagery. In *9th International Symposium of Experimental Robotics*, Singapore, June 2004.

[13] A. Rizzi and D. Koditschek. Preliminary experiments in spatial robot juggling. In *2nd.Int.Symp. Experimental Robotics*, 1991.

[14] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition*, 5:99–108, 1973.

[15] T. Soni and B. Sridhar. Modelling issues in vision based aircraft navigation during landing. In *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, pages 89–96. IEEE, 1994.

[16] M. J. van Nieuwstadt and R. M. Murray. Real time trajectory generation for differentially flat systems. *International Journal of Robust and Nonlinear Control*, 8, No. 11:995–1020, 1998.

[17] D. Zwillinger. *CRC Standard Mathematical Tables and Formulae*. CRC Press, Boca Raton, FL, 1996.