Bilateral time-scaling for control of task freedoms of a constrained nonholonomic system

Siddhartha S. Srinivasa Michael A. Erdmann Matthew T. Mason
siddh@cs.cmu.edu me@cs.cmu.edu mason@cs.cmu.edu
The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA - 15213

***Abstract*— We explore the control of a nonholonomic robot subject to additional constraints on the state variables. In our problem, the user specifies the *path* of a subset of the state variables (the *task freedoms* $\mathbf{x}_P$), i.e. a curve $\mathbf{x}_P(s)$ where $s \in [0, 1]$ is a parametrization that the user chooses. We control the *trajectory* of the task freedoms by specifying a *bilateral time-scaling* $s(t)$ which assigns a point on the path for each time t. The time-scaling is termed bilateral because there is no restriction on $\dot{s}(t)$, the task freedoms are allowed to move backwards along the path. We design a controller that satisfies the user directive and controls the remaining state variables (the *shape freedoms* $\mathbf{x}_R$) to satisfy the constraints. Furthermore, we attempt to reduce the number of control switchings, as these result in relatively large errors in our system state. If a constraint is close to being violated (at a *switching point*), we back up $\mathbf{x}_P$ along the path for a small time interval and move $\mathbf{x}_R$ to an open region. We show that there are a finite number of switching points for arbitrary task freedom paths. We implement our control scheme on the *Mobipulator* and discuss a generalization to arbitrary systems satisfying similar properties.**

I. Introduction

The goal of the Mobipulator project is to build a desktop assistant - a robot that manipulates commonplace desktop items like paper and pencil. In [1], we described the hardware and software architecture of a robot with four independantly controlled wheels, none of them steered, called the Mobipulator.

In [1], we implemented a configuration space planner that moved the paper from a start location to a goal. When the robot has two of its wheels on paper (called the *hands*) and the other two on the desktop (called the *feet*), the robot is said to be in *dual-differential drive* mode (Fig. 1). In this mode, the motion of the paper is unconstrained - the wheel velocities span the space of the paper velocities. The configuration space planner treated the paper as a trailer hitched to the center of the hands and found paths for the paper while steering the robot away from obstacles. If the dual-differential drive mode was violated, the robot moved across the paper and continued the plan.

In this paper, we explore a more dynamic task - a user controls the path of the paper remotely. We devise a controller that executes the user's directive and maintains the robot in dual-differential drive mode.

We define the state variables that the user controls as the *task freedoms* and the remaining state variables the *shape freedoms*. In our problem, the task freedoms are the pose of the paper $\mathbf{x}_P = (x_P\ y_P\ \theta_P\ )^T$ and the shape freedoms are the pose of the robot relative to the paper $\mathbf{x}_R = (x_R\ y_R\ \theta_R\ )^T$.

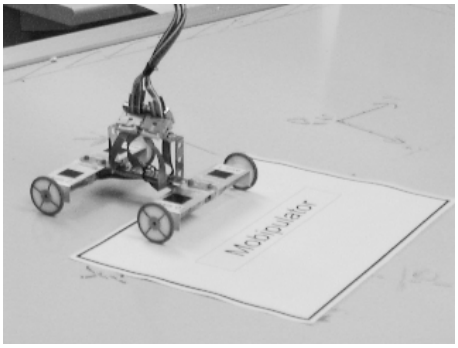

Fig. 1
DUAL-DIFFERENTIAL DRIVE MODE[1]

The decomposition of state variables into task and shape variables was used by Nakamura[2] to study robot arms. He termed the pose of the end effector as the *manipulation variable* and studied the control of joint angles to obtain desired paths.

In our problem, the user specifies the *path* of the paper, i.e. a curve $\mathbf{x}_P(s)$ in the task space where $s \in [0, 1]$ is a parametrization that the user chooses. Note that there is no notion of time in the path specification. We control the *trajectory* of the task freedoms by specifying a *time-scaling* $s(t)$ which assigns a point on the path for each $t \in [0, T]$, where T is the time to completion of the path.

The concept of time scaling was used by Bowbrow et al.[3] to find time-optimal trajectories of a fully-actuated manipulator along a specified path, subject to limitations in actuator torque. They used a time-scaling $s(t)$ with a unilateral constraint $\dot{s} > 0$, i.e. the end-effector was not allowed to move backwards along the specified path.

We will show in §4 that for our problem we cannot impose the unilateral constraint on the time-scaling function and require bilateral time-scaling with $\dot{s}$ unconstrained.

For example, in Fig. 2, the user specifies the path $(ac - cb - bd)$, we specify the time-scaling $s(t)$ and the resulting trajectory is $(ac - cb - bc - cb - bd)$.

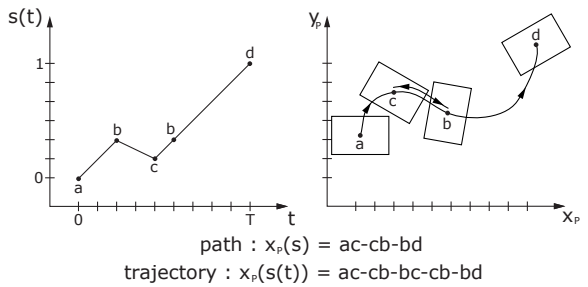

Fig. 2
TIME-SCALING OF USER'S PATH

As the user is interested only in the motion of the paper, we have an extra degree of freedom - four wheels control the three paper degrees of freedom. The configuration space planner used the extra freedom to hitch the paper to the robot. We use it to maintain the shape freedoms in dual-differential drive. Intuitively, this is similar to a moving hitch placed optimally at each instant.

The idea of using redundant degrees of freedom to optimize performance was used by Baillieul et.al.[4] in the control of redundant manipulators. They used the *extended Jacobian* technique to move the end effector along a prescribed path *and* locally optimize an objective function.

In §4, we decompose the system into the task and the shape subsystems. We rewrite the shape system as a function of the task freedoms. The user controlled task freedoms appear as a drift term in the shape system. We use bilateral time-scaling to control this drift, and the extra degree of freedom to move the shape freedoms to satisfy dual-differential drive. In §5, we provide a control policy that reduces the number of control switches required. We explain the motivation for this policy in §2. §6 describes an implementation of the control law on the real robot and some of the problems faced. §7 explores a generalization for arbitrary nonholonomic systems.

II. BACKGROUND OF NONHOLONOMIC SYSTEMS

A general form of a nonholonomic system is given by:

$$\Sigma : \qquad \dot{\mathbf{x}} = u_1 \mathbf{f_1}(\mathbf{x}) + \cdots + u_m \mathbf{f_m}(\mathbf{x}) \qquad (1)$$

where $2 \leq m < n$, $\mathbf{x} = (x_1 \cdots x_n)^T$ is the state vector defined in an open subset S of $\mathbb{R}^n$, $u_i \in \mathbb{R}$ are the control inputs, and $\mathbf{f_1}, \cdots, \mathbf{f_m}$ are vector fields on S. Σ is said to be *completely nonholonomic* if the rank of the controllability Lie algebra generated by $u_1 \cdots u_m$ is n. A completely nonholonomic system is completely controllable (*Chow's theorem*[5]).

A. Motion planning

Motion planning for nonholonomic systems is complicated by the fact that not all motions are feasible, only those which satisfy the instantaneous nonholonomic constraints. Nevertheless, the completely nonholonomic assumption guarantees that feasible motions do exist which steer an arbitrary initial state to a final state. We refer the reader to Kolmanovsky et al.[6] for a detailed review of motion planning for nonholonomic systems.

The motion planning problem for nonholonomic systems can be defined as : for every pair of points $(\mathbf{p},\mathbf{q}) \in S$, generate an open-loop control $\mathbf{u}(t) = (u_1 \cdots u_m)^T$ that steers $\mathbf{p}$ to $\mathbf{q}$.

One approach is to consider the extended system :

$$\Sigma_e : \qquad \dot{\mathbf{x}} = v_1 \mathbf{f_1}(\mathbf{x}) + \cdots + v_m \mathbf{f_m}(\mathbf{x}) + v_{m+1} \mathbf{f_{m+1}}(\mathbf{x}) + \cdots + v_r \mathbf{f_r}(\mathbf{x}) \quad (2)$$

where $\mathbf{f_{m+1}}, \cdots, \mathbf{f_r}$ are higher-order Lie brackets of the $\mathbf{f_i}$ chosen so that $\mathbf{f_1}(\mathbf{x}), \cdots, \mathbf{f_r}(\mathbf{x})$ span $\mathbb{R}^n$ for all $\mathbf{x} \in S$. (2) can be solved for the control $\mathbf{v}$(t) that steers $\mathbf{p}$ to $\mathbf{q}$.

The hard part is to generate Lie brackets from the control inputs. Fast switchings of piecewise constant or polynomial inputs[7] and high-frequency high-amplitude periodic control inputs[8] are some of the techniques used to generate motions in the directions of the Lie brackets.

B. Motions of the Mobipulator

One motion along a higher-order Lie bracket for the Mobipulator is akin to parallel parking the robot relative to the paper. This is achieved by spinning the wheels repeatedly forwards and backwards for small time intervals. We use rubber O-rings on the wheels to help better grip the paper. This also increases friction between the feet and the desktop and thus the wheels require a minimum torque before they start spinning. As a result, fast switchings of wheel torques cause a nonsmooth sideways motion and result in the wheels slipping. This produces errors in both the pose of the robot and the paper. Hence, we would like to avoid control switches for accurate motion.

III. PROBLEM STATEMENT

The kinematics for the Mobipulator in dual-diff drive mode can be described as :

$$M : \quad \begin{pmatrix} \dot{\mathbf{x}}_\mathbf{P} \\ \dot{\mathbf{x}}_\mathbf{R} \end{pmatrix} = \mathbf{A}(\mathbf{x_R}) \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} = \mathbf{A}\,\omega \qquad (3)$$

where $\mathbf{x_P} = (x_P\ y_P\ \theta_P)^T$ is the pose of the paper in the world frame, $\mathbf{x_R} = (x_R\ y_R\ \theta_R)^T$ is the pose of the robot *relative* to the paper, and the ω_i are the angular velocities of the wheels.

M requires the robot to be in dual-differential drive mode. The shaded region in Fig. 3 describes the allowable

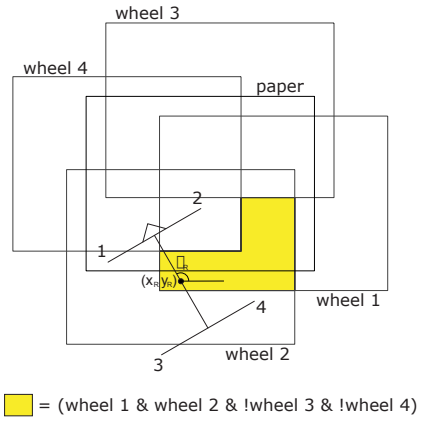

Fig. 3
ALLOWABLE $(x_R\ y_R)$ AT $\theta_R = 2\pi/3$

pose of the robot relative to the paper for a given θ_R. Each light rectangle represents the locus of the center of the robot with one of the wheels touching the edge of the paper. If the center lies in the interior of a light rectangle, the corresponding wheel lies in the interior of the paper. The shaded polygon is the locus of points with the manipulating wheels (1 and 2) on the paper and the locomoting wheels (3 and 4) on the desktop.

The allowable region depends only on the relative pose θ_R and the dimensions of the robot and the paper. The shape freedom constraints can thus be written as :

$$H: \quad \mathbf{h_1}(\theta_R) \leq \begin{pmatrix} x_R \\ y_R \end{pmatrix} \leq \mathbf{h_2}(\theta_R) \tag{4}$$

The problem can be stated as:

Given any user-specified paper path $\mathbf{x_P}(s)$, is it possible to construct wheel angular velocities ω_i that attain the specified $\mathbf{x_P}$ and satisfy the constraint H? Furthermore, is it possible to construct a control policy that reduces the number of control switches required?

IV. CONTROL OF SHAPE FREEDOMS

In this section we will prove that we can control the Mobipulator system to follow a user-specified $\mathbf{x_P}(s)$ and satisfy H. We will first consider the case where the user has control of the velocity $\dot{\mathbf{x}}_\mathbf{P}(t)$ and show that the resulting system is *not* small time locally controllable (STLC). This negative result will provide us with insight to modify the user's control to a path $\mathbf{x_P}(s)$. We will show that the resulting system is STLC only if there are no constraints on $\dot{s}(t)$.

A. The shape and task subsystems

We decompose the Mobipulator system M into two subsystems - the task system M_t and the shape system M_s. M_t comprises of the task freedoms the user controls (the $\mathbf{x_P}$) and M_s comprises of the shape freedoms (the $\mathbf{x_R}$). We will analyze each of these subsystems separately.

$$\begin{aligned} M &: \begin{pmatrix} \dot{\mathbf{x}}_\mathbf{P} \\ \dot{\mathbf{x}}_\mathbf{R} \end{pmatrix} = \mathbf{A}\,\omega = \begin{bmatrix} \mathbf{B} \\ \mathbf{C} \end{bmatrix} \omega \\ M_t &: \dot{\mathbf{x}}_\mathbf{P} = \mathbf{B}\,\omega \\ M_s &: \dot{\mathbf{x}}_\mathbf{R} = \mathbf{C}\,\omega \end{aligned}$$

B. The task subsystem

We first observe that $\mathbf{B}$ is of rank 3. This is beacause the motion of $\mathbf{x_P}$ is unconstrained in M_t, i.e. if H is satisfied, any desired $\dot{\mathbf{x}}_\mathbf{P}$ can be attained by a correct selection of the wheel velocities. We create an augmented rank 4 system by adding the nullspace vector of $\mathbf{B}$, $\mathbf{n_B}$, and an additional scalar α which we can control without affecting the task freedoms.

$$M_{ta}: \quad \begin{pmatrix} \dot{\mathbf{x}}_\mathbf{P} \\ \alpha \end{pmatrix} = \begin{bmatrix} \mathbf{B} \\ \mathbf{n_B} \end{bmatrix} \omega \tag{5}$$

We can now invert the augmented system. Intuitively, α represents the one degree of freedom that we have in choosing ω. We will use this freedom to satisfy H.

$$\omega = \begin{bmatrix} \mathbf{B} \\ \mathbf{n_B} \end{bmatrix}^{-1} \begin{pmatrix} \dot{\mathbf{x}}_\mathbf{P} \\ \alpha \end{pmatrix} \tag{6}$$

C. The shape subsystem

The shape freedoms can be rewritten in terms of the task freedoms using Eqn. 6.

$$\dot{\mathbf{x}}_\mathbf{R} = \mathbf{C} \begin{bmatrix} \mathbf{B} \\ \mathbf{n_B} \end{bmatrix}^{-1} \begin{pmatrix} \dot{\mathbf{x}}_\mathbf{P} \\ \alpha \end{pmatrix} \tag{7}$$

This can be arranged in a more intuitive form :

$$M_{sa}: \quad \dot{\mathbf{x}}_\mathbf{R} = \mathbf{F}\,\dot{\mathbf{x}}_\mathbf{P} + \mathbf{g}\,\alpha \tag{8}$$

From Eqn. 8 we can see that the motion of the task freedoms appears as a drift term $\mathbf{F}\,\dot{\mathbf{x}}_\mathbf{P}$ in the shape system and the degree of freedom α appears as the scalar control.

We can gain further insight on M_{sa} by studying $\mathbf{g}$:

$$\mathbf{g} = \frac{c}{4} \begin{pmatrix} \cos(\theta_R) \\ \sin(\theta_R) \\ 0 \end{pmatrix} \tag{9}$$

where c is the radius of each wheel.

This field lets the robot move forwards and backwards relative to the paper. For example, if there was no drift (i.e. if $\dot{\mathbf{x}}_\mathbf{P}$ were $\mathbf{0}$), then the only robot motion that does not manipulate the paper is this forwards-backwards motion.

D. A negative result

For H to be satisfied, we would like to be able to control the shape freedoms (described by M_{sa}) immaterial of the drift generated by the user-controlled task freedoms. If we allow the user to control $\dot{\mathbf{x}}_\mathbf{P}(t)$, this requires M_{sa} to be STLC for all $\dot{\mathbf{x}}_\mathbf{P}$ and $\mathbf{x_R}$. We use the following theorem to prove that this is *not* true.

Theorem 1 *The system M_{sa} is small-time locally controllable at $\mathbf{x_R}$ if and only if there exists some α^0 such that for all $\dot{\mathbf{x}}_\mathbf{P}$*

$$\mathbf{0} = \mathbf{F}\ \dot{\mathbf{x}}_\mathbf{P} + \mathbf{g}\ \alpha^0 \tag{10}$$

Proof : Refer Sussman[9]
Corollary 1 *The system M_{sa} is not STLC*
Proof : (10) is an overconstrained system with three equations and one variable. Any non-zero choice of $\dot{\mathbf{x}}_\mathbf{P}$ will yield no soulution for α^0.

Intuitively this means that the drift field will always dominate and can force the system towards a constraint boundary, and then violate it. Thus, if the user has control of $\dot{\mathbf{x}}_\mathbf{P}(t)$, it is not always possible to satisfy H.

E. Time scaling

We overcome this problem by allowing the user to control the path $\mathbf{x_P}(s)$. We control the trajectory with a time-scaling $s(t)$ that assigns a point on the path for each $t \in [0, T]$, where T is the time to completion of the path. The velocity at any time t is given by :

$$\dot{\mathbf{x}}_\mathbf{P}(\mathbf{s(t)}) = \mathbf{x}'_\mathbf{P}(\mathbf{s(t)})\ \dot{s}(t) \tag{11}$$

We control the the time-scaling rate $\dot{s}(t)$. This in turn provides us with control of the the velocity of the task freedoms along the path.

We first restrict $\dot{s}(t) > 0$, i.e. we cannot reverse the motion of the paper along the path. Denoting $\dot{s}(t)$ by β, we write the time-scaled shape system as :

$$M_{su} : \begin{pmatrix} \dot{\mathbf{x}}_\mathbf{R} \\ \dot{s} \end{pmatrix} = \begin{pmatrix} \mathbf{F}\mathbf{x}'_\mathbf{P} \\ 1 \end{pmatrix} \beta + \begin{pmatrix} \mathbf{g} \\ 0 \end{pmatrix} \alpha \tag{12}$$

M_{su} is a control-affine system with one unilateral control $\beta > 0$. We can test STLC for this system :
Theorem 2 *The system M_{su} is small-time locally controllable at $\mathbf{x_R}$ if and only if there exists some α^0 and some $\beta^0 \neq 0$ such that for all $\mathbf{x_P}$*

$$\mathbf{0} = \begin{pmatrix} \mathbf{F}\mathbf{x}'_\mathbf{P} \\ 1 \end{pmatrix} \beta^0 + \begin{pmatrix} \mathbf{g} \\ 0 \end{pmatrix} \alpha^0 \tag{13}$$

Proof : Refer Goodwine and Burdick[10]
Corollary 2 *The system M_{su} is not STLC*
Proof : The last row of (13) can be 0 for any arbitrary $\mathbf{x_P}$ only if $\beta^0 = 0$. Hence the system M_{su} is *not* STLC.

We now remove the restriction on $\dot{s}$. This results in a bilateral control β. The untestricted system M_{sb} looks like (12) but with β unrestricted. We rewrite it as :

$$M_{sb} : \quad \begin{pmatrix} \dot{\mathbf{x}}_\mathbf{R} \\ \dot{s} \end{pmatrix} = \mathbf{f}_\beta\ \beta + \mathbf{f}_\alpha\ \alpha \tag{14}$$

We prove this system is STLC with the following theorem:
Theorem 3 *The system M_{sb} is small-time locally controllable at $\mathbf{x_R}$ if the controllability Lie algebra generated iterated Lie brackets of $\mathbf{f}_\beta$ and $\mathbf{f}_\alpha$ spans the state space.*

Proof : Refer Chow[5]
Corollary 3 *The system M_{sb} is STLC*
Proof : The brackets [$\mathbf{f}_\beta$], [$\mathbf{f}_\alpha$], [$\mathbf{f}_\beta$,$\mathbf{f}_\alpha$] and [$\mathbf{f}_\beta$, [$\mathbf{f}_\beta$,$\mathbf{f}_\alpha$]] span the 4-dimensional state space for all $\mathbf{x_P} \neq \mathbf{0}$.

F. Discussion

We have proved that we can follow any task freedom path and satisfy the constraints. To show STLC, we have had to use higher order brackets of $\mathbf{f}_\beta$ and $\mathbf{f}_\alpha$. An application of these brackets will cause control switchings and the resulting motion will not be smooth. Hence we must choose our control policy in such a way that we minimize the use of the brackets. We will describe one such policy in §5. Another point of concern is that since $\dot{s}$ is unrestricted, we can move forwards and backwards along our path. We will prove in §5 that any arbitrary path can be completed in a finite amount of time.

V. Control of the *Mobipulator*

Here we will propose control policies α and β, for the Mobipulator. We will show that control switchings can be restricted to a finite number of *switching points*.

A. Control policy

Recall that the control vector field $\mathbf{g}$ (Eqn. 9) allows the motion of the robot forwards and backwards relative to the paper. This is shown as the line LL' in Fig.4. We can use α to servo to any point on this line. LL' intersects the constraint boundaries at L_{max} where the robot is farthest from paper, and L_{min} where the robot is closest to the paper. Our policy servos the robot to the midpoint L_{mid}. This maximizes the minimum distance between the wheels and the edges of the paper and is a safe policy that is less severe to motion errors.

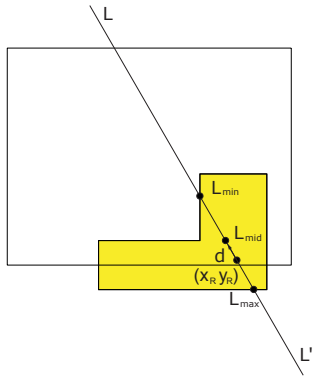

Fig. 4
Control policy for α

For β, we use the simple policy of $\beta = 1$ as long as H is satisfied. The next sub-section describes the policies chosen when H is violated.

B. Switching points

Note that with $\beta = 1$ we are deliberately allowing the paper to drift. We do this to avoid motions that require control switchings until imperative. H is violated when $L_{min} \rightarrow L_{max}$ and no further motion is possible along the control $\mathbf{g}$. This occurs either when a manipulating wheel and a locomoting wheel touch edges of the paper (Q and R in Fig.5) or when a manipulating wheel touches a corner of the paper (P in Fig.5). We term these points *switching points* as they mandate a switch in the control policy.

C. Control policy at switching points

At switching points, we move the paper backwards along the path for a small time interval δt by setting $\beta = -1$. Choice of α depends on whether we want net motion forwards or backwards. To escape from P, we need to move along x_R, while to escape from Q, we need to move along $-y_R$. Consider P. The motion of the robot along x_R is given by :

$$\dot{x}_R = \mathbf{F}(\mathbf{1})\mathbf{x}'_{\mathbf{P}}\beta + \mathbf{g}(\mathbf{1})\alpha \tag{15}$$

$\mathbf{F}(\mathbf{1})$ and $\mathbf{g}(\mathbf{1})$ are the first row of $\mathbf{F}$ and $\mathbf{g}$ respectively.

For small δt, we can approximate $\mathbf{F}(\mathbf{1})\mathbf{x}'_{\mathbf{P}}$ and $\mathbf{g}(\mathbf{1})$ as constants k_F and k_g respectively. At switching points we first set $\beta = -1$ and α to α_1. After δt, we set $\beta = 1$ and α to α_2. The net motion along x_R is given by the sum :

$$\delta x_R \approx \dot{x}_R \delta t = k_F \beta \delta t + k_g \alpha \delta t \tag{16}$$

$$\begin{aligned} \delta x_{R1} &\approx -k_F \delta t + k_g \alpha_1 \delta t \\ \delta x_{R2} &\approx k_F \delta t + k_g \alpha_2 \delta t \end{aligned}$$

$$\delta x_{R1} + \delta x_{R2} \approx k_g(\alpha_1 + \alpha_1)\delta t \tag{17}$$

By servoing to L_{max}, we can obtain the largest positive α and get the largest motion along x_R. Conversely, by servoing to L_{min}, we can obtain the largest negative α and get the largest motion along $-x_R$. Note that the net motion obtained is immaterial of the motion of the paper as k_F is cancelled during motion.

We continue the motions until the wheel is sufficiently clear of the constraining edge or vertex (defined by a threshold w shown in Fig.5). Since each of the above-mentioned motions produces the same net motion along the desired direction, w will be attained in finite time.

VI. Implementation

The Mobipulator system has a camera that tracks colored fiducials marked on the robot and the paper, and provides pose information at 10 Hz. Our test path is a hexagon whose width is 8 times the width of the robot. The paper is rotated by 60^o about its center at each vertex. Note that, with the chosen path and the starting pose of the robot, no switching points are encountered. We will describe the effect of switching points separately.

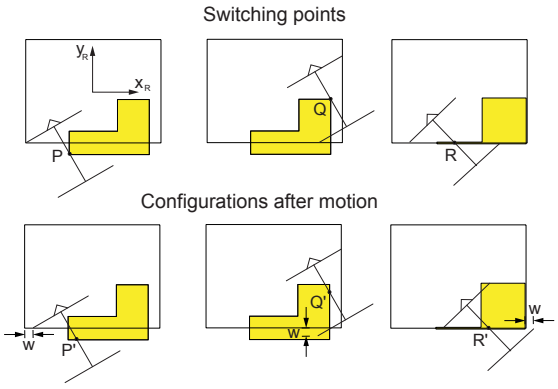

Fig. 5
Switching points

A. Open loop execution

We have written a simulator that inputs a user-specified path, executes our control policies and outputs wheel angular velocities. We fed the output to the robot. The intended path (the solid hexagon in Fig.6(b)) and two runs of the open-loop execution are shown. The path has large error because slip between the robot and the paper causes a loss of dual-differental drive. The error builds up as there is no feedback. The maximum error between the start and the goal was 16cm. and the angular error was 27^o. The robot took about 3 minutes to complete the path.

B. Error correction

Since the task system M_t is holonomic, errors in the motion of the paper can be corrected by proportional control. We used the pose information from the camera to servo the paper to its intended path. At every camera update, we computed the paper velocity required to servo to the path. This velocity was fed to the simulator which provided the necessary wheel angular velocities. We applied these velocities until the camera updated the pose again. Fig.6(c) shows four runs of the robot. The maximum deviation from the path was 4.6cm. and the maximum angular error was 4^o. The time to completion was comparable to the open-loop implementation.

C. Switching points

We ran separate trials to test the accuracy of motion at switching points. We ran 15 tests open-loop for the robot in switching point R. None of the tests were successful. This was because this motion required the wheels to be placed vey close to the edge of the paper. The accumilated slip caused a loss of dual-differential drive mode and the motion failed. When implemented with visual-servoing, 10 out of 15 tests were successful. However, the robot required 18 repetitions of the motion described in §5.3

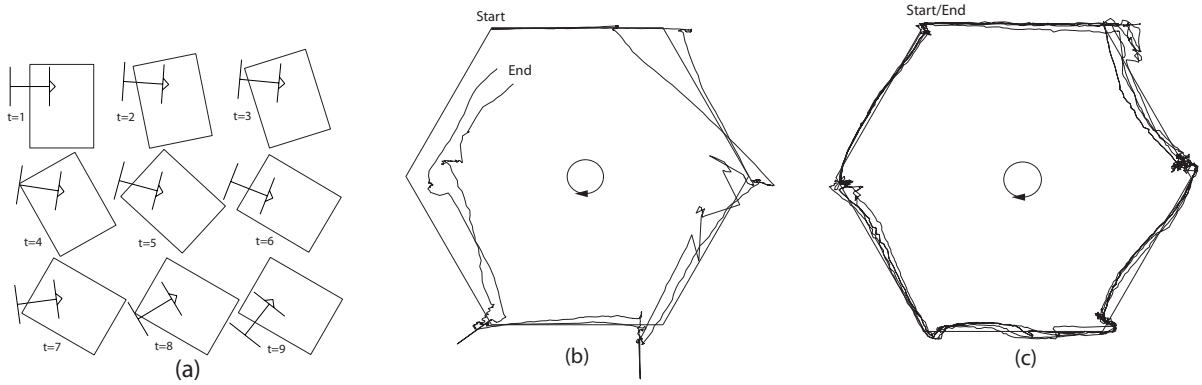

Figure 6: Implementation of the control policy to move the paper in a hexagon whos width is 8 times the width of the robot. The paper is rotated by 60^o at each vertex. (a) shows the motion of the robot during one such rotation of the paper. (b) shows the paper pose for open-loop execution. (c) shows the paper pose when the control policy is run with visual feedback.

which took about 1 minute to complete. The resulting error in the location of the paper was 1.4cm.

VII. Discussion

Our method decomposes the system state into task freedoms and shape freedoms and analyzes each subsystem separately. By doing so, we reduce a nonholonomic control problem to two subproblems - one of following a holonomic path in task space and another of controlling the nonholonomic shape freedoms to satisfy constraints. We thus isolate the nonholonomy to a smaller set of state variables. By designing a suitable control policy in shape space we can also ensure that the task space path is followed smoothly with reduced control switchings. Furthermore, since the task system is holonomic, error correction is easy.

The method outlined in $\S 4$ can be applied to any nonholonomic system. The key lies in the fact that the matrix $\mathbf{B}$ is full rank. Given a nonholonomic system that looks like M, we can perform row operations on the $\mathbf{A}$ and generate a full rank $\mathbf{B}$. The corresponding state variables can be treated as the task freedoms and the remaining state variables can be treated as the shape freedoms. We can then apply the same theorems outlined in $\S 4$ to prove STLC and generate control laws.

VIII. Future Work

We will work on policies and paths that minimize control switchings. Currently, we use the safe policy of using α to servo to L_{mid}. One can imagine a mixed policy that servoes the robot to L_{max} when the robot is close to a corner to increase the clearance between the robot and the paper. It is unclear as to whether there exists a single optimal policy for all possible paths. Another interesting problem arises when the controller has the freedom to choose the path - given a fixed policy, what is the path that minimizes the number of switching points.

IX. Acknowledgements

We thank Rashmi Patel for help with the implementation. This work was supported in part by NSF grants IIS-9820180, IIS-9900322, IIS-0082339 and IIS-0222875.

X. References

[1] Srinivasa, Baker, Sacks, Reshko, Mason, and Erdmann, "Experiments with nonholonomic manipulation," in *IEEE Int. Conf. on Rob. and Auto.*, 2001.

[2] Y. Nakamura, "Advanced robotics-redundancy and optimization," in *Addison-Wesley*, 1991.

[3] J.Bowbrow, S.Dubowsky, and J.Gibson, "Time-optimal control of robot manipulators along specified paths," *Int. J. of Rob. Research*, vol. 4, 1985.

[4] J.Baillieul, "Kinematic programming alternatives for redundant manipulators," in *IEEE Int. Conf. on Robotics and Automation*, 1985, pp. 722–728.

[5] W.Chow, "Uber systeme von linearen partiallen differentialgleichungen erster ordnung," *Math Ann.*, vol. 117, pp. 98–105, 1939.

[6] I.Kolmanovsky and N.McClamroch, "Developments in nonholonomic control problems," in *IEEE Control systems magazine*, 1995, pp. 20–36.

[7] G. Lafferriere and H. Sussmann, "A differential geometric approach to motion planning," in *Nonholonomic Motion Planning*, 1993, pp. 235–270.

[8] H. Sussmann and W.Liu, "Limits of highly oscillatory controls and approximation of general paths by admissible trajectories," in *IEEE International conference on decision and control*, 1991.

[9] H.Sussman, "Lie brackets and local controllability : A sufficient condition for local scalar-input systems," *SIAM J. on control and opt.*, vol. 21(5), 1978.

[10] B. Goodwine and J. Burdick, "Controllability with unilateral control inputs," in *IEEE Conference on Decision and Control*, 1996.