

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

Siddhartha S. Srinivasa, Michael A. Erdmann, Matthew T. Mason
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 $t \in [0, T]$, where T is the time to completion of the path. 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) such that the constraints are satisfied. 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}_S 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.

1 Introduction

The goal of the Mobipulator project is to build a desktop assistant - a robot that manipulates common-place desktop items like paper and pencil. In [1], we described the hardware and software architecture of a robot with four independently 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 until the feet were on the paper and the hands were on the desktop and continued the plan. The trailer-hitch assumption simplified planning, and we were able to generate fast plans.

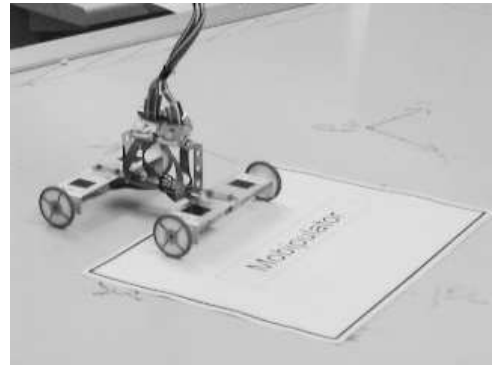


Figure 1: Dual-differential drive mode[1]

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 $\mathbf{x}_R = (x_R y_R \theta_R)^T$.

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 con-

trol of joint angles to obtain desired paths. Ostrowski and Burdick[3] studied undulatory locomotion of the *Snakeboard* where motion was generated by coupling internal shape changes with external nonholonomic constraints.

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.[4] 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 $(ac - cb - bc - cb - bd)$.

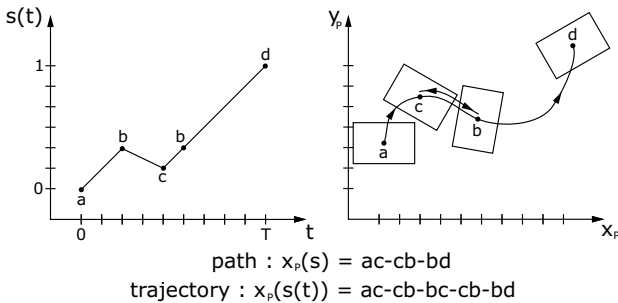


Figure 2: Time-scaling of user's path

Since we are only interested in the motion of the paper, we have an extra degree of freedom - there are four wheels controlling the three paper degrees of freedom. The configuration space planner used the extra degree of freedom to hitch the paper to the robot. We use the extra degree of freedom to control the shape freedoms to maintain the robot in dual-differential drive mode. 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.[5] 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. They used one such function, the *manipulability index*, to steer the manipulator away from kinematic singularities.

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. We then explore a generalization for arbitrary nonholonomic systems with similar properties.

2 Background of nonholonomic systems

A general form of a nonholonomic system is given by the drift-free nonlinear control system :

$$\Sigma : \quad \dot{\mathbf{x}} = u_1 \mathbf{f}_1(\mathbf{x}) + \dots + u_m \mathbf{f}_m(\mathbf{x}) \quad (1)$$

where $2 \leq m < n$, $\mathbf{x} = (x_1 \dots x_n)^T$ is the state vector defined in an open subset S of \mathbb{R}^n , $u_i \in \mathbb{R}$ are the control inputs, and $\mathbf{f}_1, \dots, \mathbf{f}_m$ are vector fields on S .

Σ is said to be *completely nonholonomic* if the rank of the controllability Lie algebra generated by $u_1 \dots u_m$ is n . A completely nonholonomic system is completely controllable (*Chow's theorem*[6]).

2.1 Motion planning

Motion planning for nonholonomic systems is complicated by the fact that not all motions are feasible, only those motions which satisfy the instantaneous nonholonomic constraints. Nevertheless, the completely nonholonomic assumption guarantees that feasible motions do exist which steer an arbitrary initial state to an arbitrary final state. Kolmanovsky et al.[7] contains a detailed review of motion planning for nonholonomic systems. We state here a sampling of the literature to give the reader a flavor of the approaches used.

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 :

$$\Sigma_e : \quad \dot{\mathbf{x}} = v_1 \mathbf{f}_1(\mathbf{x}) + \cdots + v_m \mathbf{f}_m(\mathbf{x}) + v_{m+1} \mathbf{f}_{m+1}(\mathbf{x}) + \cdots + v_r \mathbf{f}_r(\mathbf{x}) \quad (2)$$

where $\mathbf{f}_{m+1}, \dots, \mathbf{f}_r$ are higher-order Lie brackets of the \mathbf{f}_i chosen so that $\mathbf{f}_1(\mathbf{x}), \dots, \mathbf{f}_r(\mathbf{x})$ 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[8] and high-frequency high-amplitude periodic control inputs[9] are some of the techniques used to generate motions in the directions of the Lie brackets.

2.2 Motions of the Mobipulator

One example of a motion along a higher-order Lie bracket for the Mobipulator system is akin to parallel parking the robot relative to the paper. This can be achieved by spinning the wheels repeatedly forwards and backwards for small time intervals, and produces a sideways motion of the robot. We use rubber O-rings on the wheels to help better grip the paper. This also increases the friction between the feet and the desktop and thus the wheels require a threshold 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 such control switches for accurate motion.

2.3 Motion planning with obstacle avoidance

Gurvits and Li[10] proposed an approach for non-holonomic motion planning in the presence of obstacles. Their method first constructed a path connecting the start and the goal that avoided obstacles but was not feasible. They then used high-frequency high-amplitude periodic inputs to generate an approximate path.

Mirtich and Canny[11] used skeletons - collections of fixed (typically nonfeasible) paths which stay maximally clear of the obstacles - to steer a mobile robot from an initial to a final state. Low complexity paths were generated by making the system loosely follow the skeleton. Jacobs et al.[12] developed a different approach, also based on approximating a nonfeasible path by feasible path segments, to plan motions for a

car-like robot.

Barraquand and Latombe[13] used a potential field method to produce collision-free paths for mobile robots. Their planner applied constant controls for a small time interval to generate discrete actions. These actions were used to generate plans that minimized time and the number of switches in the controls.

3 Problem statement

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

$$M : \quad \begin{pmatrix} \dot{\mathbf{x}}_P \\ \dot{\mathbf{x}}_R \end{pmatrix} = \mathbf{A}(\mathbf{x}_R) \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} = \mathbf{A} \omega \quad (3)$$

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.

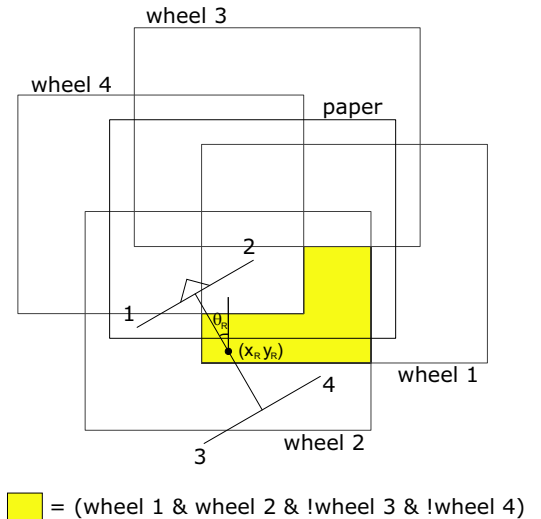


Figure 3: Allowable $(x_R \ y_R)$ at $\theta_R = \pi/6$

The Mobipulator system M requires the robot to be in dual-differential drive mode. The shaded polygon in Fig. 3 describes the allowable pose of the center 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 constraints on the shape freedoms can thus be written as :

$$H : \mathbf{h}_1(\theta_R) \leq \begin{pmatrix} x_R \\ y_R \end{pmatrix} \leq \mathbf{h}_2(\theta_R) \quad (4)$$

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?

As explained in §2.2, we would like to avoid control switches as these produces motions that cause large errors in the state variables.

4 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 $\dot{\mathbf{x}}_P(t)$ 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 $\dot{\mathbf{x}}_P(s)$. We will show that the resulting system is STLC only if there are no constraints on $\dot{s}(t)$.

4.1 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 $\dot{\mathbf{x}}_P$) and M_s comprises of the shape freedoms (the $\dot{\mathbf{x}}_R$). We will analyze each of these subsystems separately.

$$\begin{aligned} M & : \begin{pmatrix} \dot{\mathbf{x}}_P \\ \dot{\mathbf{x}}_R \end{pmatrix} = \mathbf{A} \omega = \begin{bmatrix} \mathbf{B} \\ \mathbf{C} \end{bmatrix} \omega \\ M_t & : \dot{\mathbf{x}}_P = \mathbf{B} \omega \\ M_s & : \dot{\mathbf{x}}_R = \mathbf{C} \omega \end{aligned}$$

4.2 The task subsystem

We first observe that \mathbf{B} is of rank 3. This is because the motion of \mathbf{x}_P is unconstrained in M_t , i.e. if H is satisfied, any desired $\dot{\mathbf{x}}_P$ 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.

$$M_{ta} : \begin{pmatrix} \dot{\mathbf{x}}_P \\ \alpha \end{pmatrix} = \begin{bmatrix} \mathbf{B} \\ \mathbf{n}_B \end{bmatrix} \omega \quad (5)$$

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

$$\omega = \begin{bmatrix} \mathbf{B} \\ \mathbf{n}_B \end{bmatrix}^{-1} \begin{pmatrix} \dot{\mathbf{x}}_P \\ \alpha \end{pmatrix} \quad (6)$$

4.3 The shape subsystem

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

$$\dot{\mathbf{x}}_R = \mathbf{C} \begin{bmatrix} \mathbf{B} \\ \mathbf{n}_B \end{bmatrix}^{-1} \begin{pmatrix} \dot{\mathbf{x}}_P \\ \alpha \end{pmatrix} \quad (7)$$

This can be arranged in a more intuitive form :

$$M_{sa} : \dot{\mathbf{x}}_R = \mathbf{F} \dot{\mathbf{x}}_P + \mathbf{g} \alpha \quad (8)$$

From Eqn. 8 we can see that the motion of the task freedoms appears as a drift term $\mathbf{F} \dot{\mathbf{x}}_P$ 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 the control vector field \mathbf{g} .

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

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 the desired paper velocity $\dot{\mathbf{x}}_P$ was $\mathbf{0}$), then the only robot motion possible that does not manipulate the paper is this forwards-backwards motion.

4.4 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 $\dot{\mathbf{x}}_P(t)$, this requires M_{sa} to be STLC for all $\dot{\mathbf{x}}_P$ and \mathbf{x}_R . We use the following theorem for single-input systems 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 $\dot{\mathbf{x}}_P$*

$$\mathbf{0} = \mathbf{F} \dot{\mathbf{x}}_P + \mathbf{g} \alpha^0 \quad (10)$$

Proof : Refer Sussman[14]

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 solution 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 .

4.5 Time scaling

We overcome this problem by allowing the user to control the path $\dot{\mathbf{x}}_{\mathbf{P}}(s)$. 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 :

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

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 :

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

M_{su} is a control-affine system with one unilateral control $\beta > 0$. We test STLC for this system with the following theorem :

Theorem 2 *The system M_{su} is small-time locally controllable at $\mathbf{x}_{\mathbf{R}}$ if and only if there exists some α^0 and some $\beta^0 \neq 0$ such that for all $\mathbf{x}_{\mathbf{P}}$*

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

Proof : Refer Goodwine and Burdick[15]

Corollary 2 *The system M_{su} is not STLC*

Proof : We observe that the last row of (13) can be 0 for any arbitrary $\mathbf{x}_{\mathbf{P}}$ 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 unrestricted system M_{sb} looks like (12) but with no restrictions on β . For ease

of notation, we write it as :

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

We prove that this system is STLC by using the following theorem :

Theorem 3 *The system M_{sb} is small-time locally controllable at $\mathbf{x}_{\mathbf{R}}$ if the controllability Lie algebra generated iterated Lie brackets of \mathbf{f}_{β} and \mathbf{f}_{α} spans the state space.*

Proof : Refer Chow[6]

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 $\mathbf{x}_{\mathbf{P}} \neq \mathbf{0}$.

4.6 Discussion

We have proved that we can follow any user-specified task freedom and satisfy the constraints. To show that the shape freedoms are STLC, we have had to use higher order brackets of \mathbf{f}_{β} and \mathbf{f}_{α} . We have seen that an application of these brackets will cause a switching of the control variables and the resulting motion will not be smooth. Hence we must be careful in choosing our control policies α and β to ensure that we avoid using the brackets whenever possible. We will describe one such policy in §5. Another point of concern is that \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.

5 Control of the Mobipulator

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

5.1 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. There are several policies that we can choose from. 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. Servoing to L_{max} ensures that there is maximal clearance between the robot and the paper, thereby reducing the chance

of the paper intersecting the back wheels while turning. Servoing to L_{min} brings the robot closest to the paper and might be useful in avoiding obstacles on the desktop. However, both these policies require one of the wheels to touch the edge of the paper and are not robust to errors in robot motion.

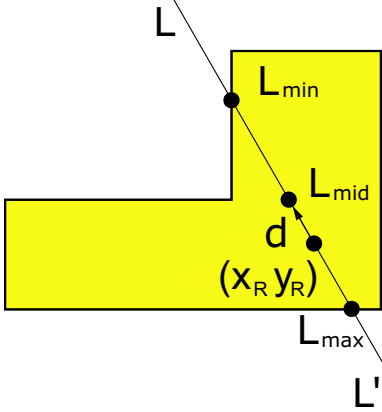


Figure 4: Control policy for α

The policy that we use servos the robot to the midpoint of its allowed motion, 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. The policy can be written as :

$$\alpha = k_p d \quad (15)$$

where d is the signed distance between (x_R, y_R) and L_{mid} , and k_p is the gain.

For β , we use the simple policy of $\beta = 1$ as long as the constraints are satisfied. We will describe the situations at which the constraints are violated, and explain the new policies at those points.

5.2 Switching points

We note that with the control policy $\beta = 1$, we have no control over the evolution of the constraint region, we are deliberately allowing the paper to drift. Our motivation for doing this is to minimize motions that require control switchings until imperative.

The constraints H are violated when $L_{min} \rightarrow L_{max}$ and no further motion is possible along the control vector field \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 as *switching points* as they mandate

a switch in the control policy. Note that P and Q occur when the robot reaches a vertex of the constraint polygon. R occurs when two opposite edges of the constraint polygon converge.

5.3 Control policy at switching points

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

$$\dot{x}_R = \mathbf{F}(\mathbf{1})\mathbf{x}'_P\beta + \mathbf{g}(\mathbf{1})\alpha \quad (16)$$

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

For small δt , we can approximate $\mathbf{F}(\mathbf{1})\mathbf{x}'_P$ and $\mathbf{g}(\mathbf{1})$ as constant 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 . Thus the net motion along x_R is given by the sum of the motions in these two steps δx_{R1} and δx_{R2}

$$\delta x_R \approx \dot{x}_R \delta t = k_F \beta \delta t + k_g \alpha \delta t \quad (17)$$

$$\delta x_{R1} \approx -k_F \delta t + k_g \alpha_1 \delta t$$

$$\delta x_{R2} \approx k_F \delta t + k_g \alpha_2 \delta t$$

$$\delta x_{R1} + \delta x_{R2} \approx k_g (\alpha_1 + \alpha_2) \delta t \quad (18)$$

By servoing to L_{max} , we can obtain the largest positive α at each instant of the motion and thereby get the largest motion along x_R . Conversely, by servoing to L_{min} , we can obtain the largest negative α at each instant and get the largest motion along $-x_R$. An important point to note that the net motion obtained is immaterial of the motion of the paper as the k_F is cancelled by the forwards and backwards motion.

We continue the motions until the wheel is sufficiently clear of the constraining edge or vertex (defined by a threshold w on the acceptable distance). Since each of the abovementioned motions produces the same net motion along the desired direction, this threshold w will be attained in a finite amount of time. Fig.5 shows the final configurations at the end of the bracketing motion. At configurations P' , Q' or R' , the rear wheels are clear of the paper and robot can swing the back wheels around the vertex and escape the constraint.

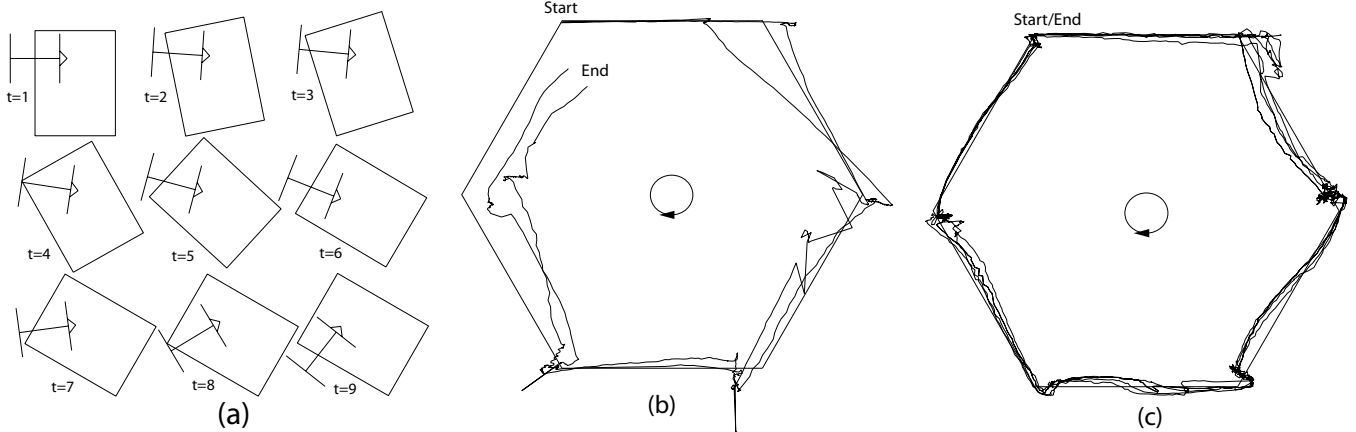


Figure 6: Implementation of the control policy to move the paper in a hexagon whose width is 8 times the width of the robot. The paper is rotated by 60° at each vertex. (a) shows the motion of the robot during one such rotation of the paper. (b) shows the locus of paper pose when the output from the simulator is run open-loop on the robot. (c) shows the locus of paper pose when the control policy is run with visual feedback from the camera.

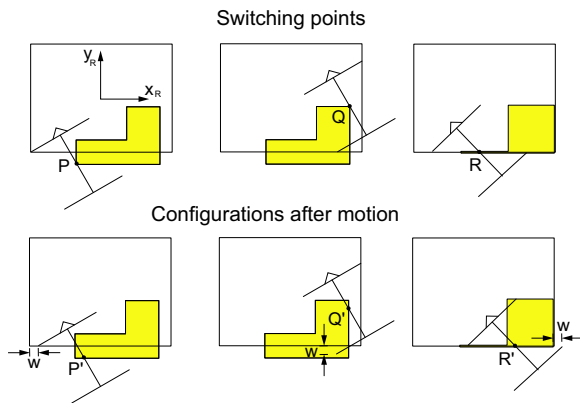


Figure 5: Switching points

6 Implementation

In this section, we describe the implementation of the control policy on the Mobipulator. The system has a Sony DFW-VL500 Firewire camera that overlooks the desktop. We use the camera both for data-logging and for visual servoing. The camera 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. This width (56cm) was chosen so that the entire path would be seen by the camera. The paper is rotated by 60° 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.

6.1 Open loop execution

We have written a simulator that takes a user-specified path as input, executes the control policies described in §5 and outputs wheel angular velocities. We fed the output of the simulator to the robot. The camera was used for data-logging and not for feedback. The intended path (the solid hexagon in Fig.6(b)) and two runs of the open-loop execution are shown. The path has a large error because slip between the robot and the paper causes a loss of dual-differential drive mode. Once out of this mode, the motion of the paper is unpredictable and results in error. This error builds up as there is no feedback. The maximum error between the start and the goal was observed to be 16cm. and the angular error was 27° . The robot took about 3 minutes to complete the path.

6.2 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. The camera updates pose at 10Hz. At every 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 closed loop execution. The maximum deviation from the path was 4.6cm. and the maximum angular error was 4° . The time taken to complete the path was comparable to the open-loop implementation.

6.3 Switching points

We ran separate tests to test the accuracy of motion at switching points. We ran 15 tests open-loop for the robot stuck in switching point R and none of the tests were successful. This was because this motion required the wheels to be placed very close to the edge of the paper. The accumulated 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 which took about 1 minute to complete. The resulting error in the location of the paper was 1.4cm. Note that, as described in §6.2, this error can be easily corrected for.

7 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 have thus isolated the nonholonomy to a smaller set of state variables. By designing a control policy in shape space that minimizes the occurrence of switching points, we can also ensure that the task space path is followed smoothly and reduced control switchings. Furthermore, since the task system is holonomic, deviations from the prescribed path can be easily corrected.

The method outlined in §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 §4 to prove STLC and generate control laws.

8 Future Work

In the future, we will work on policies and paths that reduce the number of switching points. In our current implementation, we use the safe policy of using α to servo to L_{mid} . One can imagine a policy that combines safety and also tries to maximize the time spent in dual-differential drive mode. An example is a policy that serves 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 moves the paper from a start to a goal while minimizing the number of switching points.

9 Acknowledgements

We would like to thank Rashmi Patel for his help in implementing the algorithm on the Mobipulator. This work was supported in part by NSF grants IIS-9820180, IIS-9900322, IIS-0082339 and IIS-0222875.

References

- [1] S.Srinivasa, C.Baker, E.Sacks, G.Reshko, M.Mason, and M.Erdmann, "Experiments with mobile manipulation," in *IEEE Int. Conf. on Robotics and Automation*, 2001.
- [2] Yoshihiko Nakamura, "Advanced robotics-redundancy and optimization," in *Addison-Wesley publishing company*, 1991.
- [3] J.Ostrowski and J.Burdick, "Gait kinematics for a serpentine robot," in *IEEE Int. Conf. on Robotics and Automation*, 1996, pp. 1294–9.
- [4] J.Bowbrow, S.Dubowsky, and J.Gibson, "Time-optimal control of robot manipulators along specified paths," *Int. Journal of Robotics Research*, vol. 4, 1985.
- [5] J.Baillieul, "Kinematic programming alternatives for redundant manipulators," in *IEEE Int. Conf. on Robotics and Automation*, 1985, pp. 722–728.
- [6] W.Chow, "Uber systeme von linearen partialen differentialgleichungen erster ordnung," *Math Ann.*, vol. 117, pp. 98–105, 1939.
- [7] I.Kolmanovsky and N.McClamroch, "Developments in nonholonomic control problems," in *IEEE Control systems magazine*, 1995, pp. 20–36.
- [8] G. Lafferriere and H. Sussmann, "A differential geometric approach to motion planning," in *Nonholonomic Motion Planning*, Kluwer, 1993, pp. 235–270.
- [9] 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, pp. 437–442.
- [10] L.Gurvits and Z.Li, "Smooth time-periodic feedback solutions for nonholonomic motion planning," in *Nonholonomic Motion Planning*, Kluwer, 1993.

- [11] B.Mirtich and J.Canny, “Using skeletons for nonholonomic path planning among obstacles,” in *IEEE Int. conf. on robotics and automation*, 1992.
- [12] P. Jacobs, J-P Laumond, and M.Taix, “A complete iterative motion planner for a car-like robot,” in *Journées Geometrie Algorithmique, INRIA*, 1990.
- [13] J.Barraquand and J-C Latombe, “Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles,” *Algorithmica*, vol. 10, pp. 121–155, 1993.
- [14] H.Sussman, “Lie brackets and local controllability : A sufficient condition for local scalar-input systems,” *SIAM J. on control and opt.*, vol. 21, pp. 686–713.
- [15] B. Goodwine and J. Burdick, “Controllability with unilateral control inputs,” in *IEEE Conference on Decision and Control*, 1996.