Robotica (2001) volume 19, pp. 395–405. Printed in the United Kingdom © 2001 Cambridge University Press

A real-time trajectory modification algorithm

Vadim Rogozin,* Yael Edan† and Tamar Flash*

(Received in Final Form: February 14, 2001)

SUMMARY

This paper presents a real-time algorithm for modifying the trajectory of a manipulator approaching a moving target. The algorithm is based on the superposition scheme; a model developed based on human motion behavior. The algorithm generates a smooth trajectory toward the new target by calculating the vectorial sum between the first trajectory (initial position and first target) and second trajectory (between first and second target location). The algorithm searches for the switch hme that will result in a minimum time trajectory. The idea of the algorithm is to define some domain where the optimal switching time can be found, reduce this domain as much as possible to decrease the number of the points that must be checked and try every remaining candidate in this domain to find numerically the best (optimal) switch time. The algorithm was implemented on an Adept-one robotic system taking into account velocity constraints. The actual velocity profile was found to be less smooth than specified by the mathematical model. When the switch occurs at the middle of the trajectory when the speed is close to its maximum, the change in the movement direction is performed more gently.

KEYWORDS: Trajectory modification; Moving target; Adept-one robotic system

1. INTRODUCTION

Many robot applications require operation under dynamically changing conditions. For example, in object tracking, the goal of the manipulator is to grasp a moving object. The object position continuously changes and the manipulator must update its trajectory according to changes in the object position. Another example of a need requiring to dynamically update the manipulator's trajectory is when only partial information about the target location is known a priori, or when the available information is not accurate enough. If the exact position of the target can be determined only after the movement has begun, the robot must begin its motion without knowledge of the final end-point of the trajectory and must update its trajectory once it receives additional information about the target location. The change in target position either due to the physical movement of the target or due to additional positional information provided during the motion is denoted here as a *target switch*.

All above examples require continuous updating of an ongoing trajectory due to possible target switches. Such trajectory re-planning must be performed very quickly, "on the fly" without interrupting the current movement. Updating the trajectory while the movement is in progress is termed *trajectory modification*.

The problem of trajectory modification has been solved using different approaches and mathematical tools.[1–4] Every method has its own advantages and shortcomings. The objective of this work was to design a real-time algorithm for trajectory modification based on the superposition scheme.[5] The superposition scheme, fully described in the next section, is a trajectory modification algorithm, which generates a smooth trajectory that minimizes the duration of the movement. Takayama and Kano[6] have developed a formal method of motion planning for robotic manipulators based on the concept of unit motions and the synthesis of more complex trajectories from the superposition of time-shifted and weighted unit motions. However, the focus of that study was on developing the notion of operation of motions. Hence, no algorithm was suggested in that study for choosing the appropriate temporal parameters for the motion units used in the trajectory planning scheme.

The real-time algorithm developed for the superposition strategy is presented in Section 3. The idea of the algorithm is to find a domain where the optimal switching time to the new target can be found, reduce this domain as much as possible to decrease the number of the points that must be checked and try every remaining candidate in this domain to find numerically the best (optimal) switching time. This approach can also augment feedback control, whereby the superposition scheme can be used for on-line control and not only for planning desired motions as was recently demonstrated in reference [7] whereby the superposition scheme was used for on-line tracking of a moving object by a robot manipulator. In Section 4 details of the implementation of this algorithm on an industrial robotic system are presented. Experimental results are presented in Section 5. Section 6 discusses the conclusions as well as possible suggestions for future research.

2. THE SUPERPOSITION STRATEGY

2.1. Background

The goal of the trajectory modification algorithm is to modify the robot's current trajectory and to plan a new

* Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100 (Israel). E-mail: tamar@weizmann
† Dept. of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer Sheva 841054 (Israel). E-mail: yael@bgumail.bgu.ac.il

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press

trajectory so the robot can reach the new target's location. The whole trajectory from the beginning of the movement to its end must satisfy given acceleration and/or velocity constraints. In addition, the total time of motion should be the minimal possible one. One possible way to change the trajectory is to abort the rest of the initially planned trajectory and start a new one toward the new target location. The switching between the two trajectories should be as smooth as possible. An alternative way to deal with this problem is to incorporate in some way the old trajectory into a new one toward the new location of the target. The superposition strategy is based on results from studies of human motor behavior[8,9] in target-tracking tasks. In these studies the target was first presented an initial position of the target which was then switched to a new location. A model that suggests that the modified trajectory results from the superposition of two trajectories described the trajectory of human subjects in these experiments. The trajectory between the initial location and the first target location is planned based on a displacement vector connecting them and is not aborted even after the change in target location but is performed entirely from its beginning to its end. When the target location is changed the trajectory between the first and second target locations is computed upon the appearance of the second target. The planning of this second unit of motion is based on the displacement vector from the first and the second target location. The second unit is performed by the end-effector simultaneously and independently of the first one. Thus, the actual trajectory results with the vectorial sum of the two trajectory units with an appropriate delay between them. In addition, each unit was found to be a minimum jerk trajectory.[8] This superposition strategy has three important advantages that makes it suitable for robotic applications:[5]

(i) The modification of the trajectory can take place and be completed independently of the time of the target switch. It is not necessary to know the hand's location, velocity and acceleration at the time of the trajectory modification, since it is not required for the computation of the new trajectory,
(ii) The basic superimposed trajectory units can be planned in parallel. Therefore, to plan the elementary motions it is possible to use a multi processor system. This enables high degree of parallelism both in the planning and in carrying out the task,
(iii) If both trajectories are smooth, so is their sum and the whole motion is both smooth and achieves the desired values of end position, velocity and acceleration.

2.2. The mathematical model

Let *Start*, *End* and *New* be the initial, end and new targets, respectively. The movement starts from the *Start* target to the *End* one and at some intermediate point in time the *New* target appears. At this time, the *End* target becomes irrelevant and the current start-to-end movement must be changed to ensure that the robot arm intercepts the *New* target.

According to the superposition strategy[10] the vectorial sum of two trajectories must be computed: the *Start-to-End* trajectory and *End-to-New* trajectory. The sum should give a smooth trajectory which will lead to the *New* target without violating the given constraints. However, if both trajectories satisfy certain constraints, this does not imply that the vectorial sum satisfies the constraints. Hence, it is possible that the summed trajectory may violate the given velocity or acceleration constraints even though the individual trajectories do not violate these constraints. Besides, even if the summed trajectory does not violate the constraints it might not be the minimum-time trajectory. Thus, the problem is to find the time of switching from the original trajectory to the new one S which provides a trajectory that minimizes the total movement time without violating the kinematic constraints.

Let T_1 be the time it takes to move from *Start* to *End* and T_2 be the time of movement from *End* to *New*. Let also S_1 be the time when we are informed that the *New* target has appeared and S be the time when the switch actually occurs.

In the study involving biological movement[5] each unit was found to be a minimum jerk trajectory minimizing the integrated squared hand jerk between two end points which can be adequately described by the following 5-th order polynomial:

$$X(t) = X_0 + (X_f - X_0) \cdot \left[6\left(\frac{t}{T_1}\right)^5 - 15\left(\frac{t}{T_1}\right)^4 + 10\left(\frac{t}{T_1}\right)^3 \right]$$

where T_1 is the duration of the movement.

This expression satisfies the initial position X_1, zero initial velocity and acceleration, zero final velocity and acceleration and final position X_f. For the second movement that starts at some time S and has zero initial position, velocity and acceleration, the expression becomes:

$$X(t) = A_2 \cdot \left[6\left(\frac{t-S}{T_2}\right)^5 - 15\left(\frac{t-S}{T_2}\right)^4 + 10\left(\frac{t-S}{T_2}\right)^3 \right]$$

where A_2 is the amplitude of the movement.

The velocity of the movement can be derived by differentiating the above polynomials:

$$V_1(t) = X_1'(t) = A_1 \cdot \left[30\left(\frac{t}{T_1}\right)^4 - 60\left(\frac{t}{T_1}\right)^3 + 30\left(\frac{t}{T_1}\right)^2 \right]$$

$$V_2(t) = X_2'(t) = A_2 \cdot \left[30\left(\frac{t-S}{T_2}\right)^4 - 60\left(\frac{t-S}{T_2}\right)^3 + 30\left(\frac{t-S}{T_2}\right)^2 \right]$$

Each motion unit has a "bell-shaped" velocity profile. The summation of two motion units with such velocity profiles results in the profile represented in Figure 1. In the figure the solid line marks the summed trajectory and the dashed line marks the separate trajectory units. Both units have equal amplitudes, S is approximately equal to $\frac{1}{2}T_1$.

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press

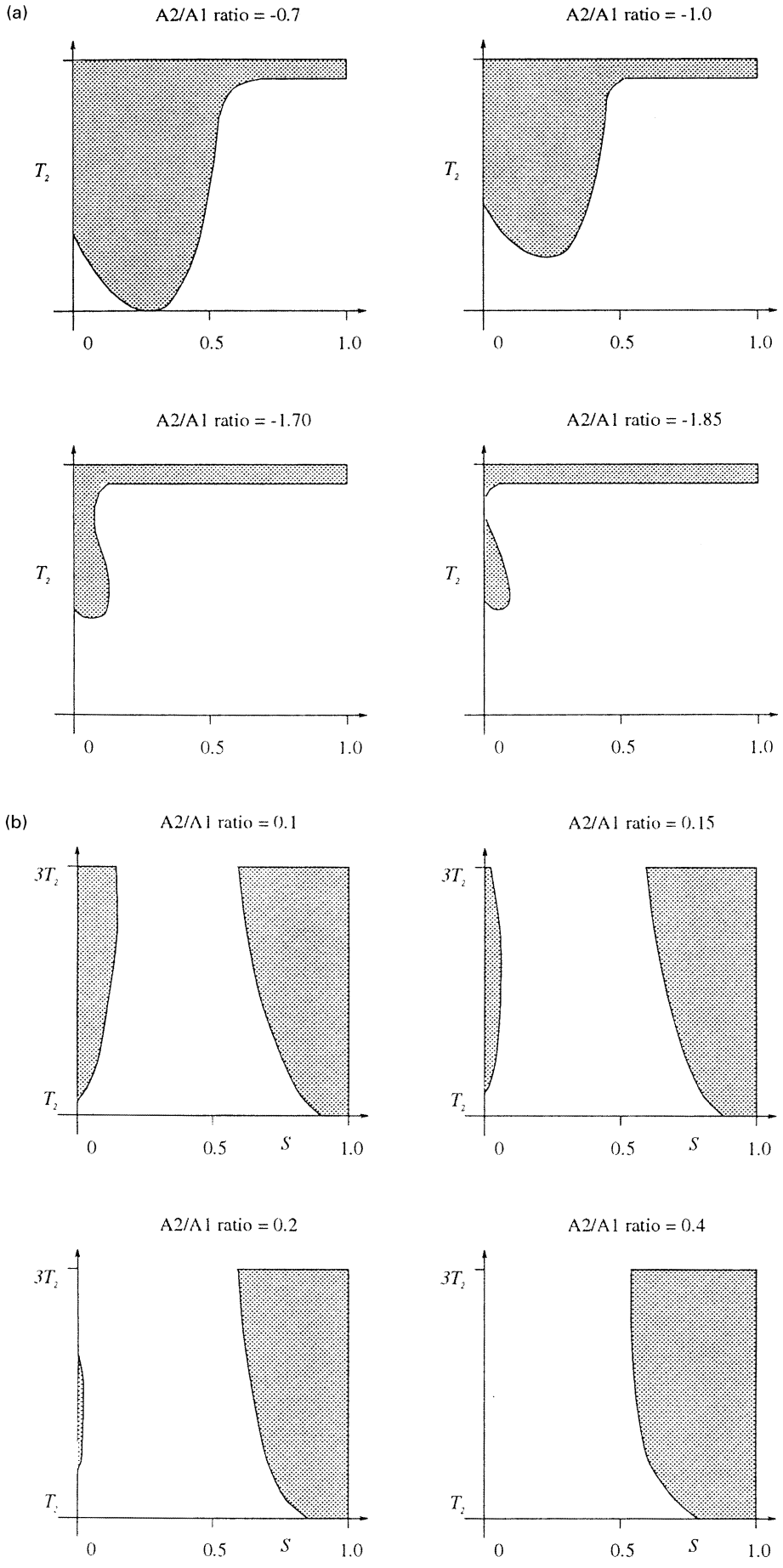


Fig. 1. RTS domain for different A_2/A_1 ratios.

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press

The goal of the algorithm presented here is to find T_2 and S that satisfy:

$$\max\left(\max_{(T_2,\,S)}(T_2+S),\,T_1\right)$$

where:

$$(T_2,\,S)\in\{[0,\,\infty)\times[S,\,\infty)\,|\,\mathrm{Check}(S,\,T_2)=\mathrm{TRUE}\}$$

Here Check(S, T_2) is a check-function which has the value TRUE, if the sum of the frst trajectory and the second one, summed at the switching point S with time T_2 for the second trajectory does not violate constraints, and FALSE otherwise. This function implicitly incorporates the given velocity and/or acceleration constraints. In this paper the superposition algorithm has been developed and implemented for the case of velocity constraints. However, the suggested method can be used for any type of constraints incorporated into the check-function.

3. A REAL-TIME ALGORITHM FOR THE SUPERPOSITION STRATEGY

For real-time application, the superposition algorithm must be fast enough because re-calculation of the trajectory is usually performed "on the fly", without interrupting the robotic motion. Previous work[5] indicated the feasibility of applying the superposition strategy for robotic control. However, the search for the optimal points proved to be too time-consuming and not applicable for real-time systems.

In the following, a new fast algorithm for searching for the optimal S and T_2 is developed. The idea of the algorithm is to define some domain where the optimal switching time S can be found, reduce this domain as much as possible to decrease the number of points checked and try every remaining candidate in this domain to find numerically the best (optimal) S.

Analytical solution of the problem by applying optimal control theory was also considered. However, the derived non-linear equations required complicated numerical procedures tor their solution and therefore were not feasible for real-time implementation.

It was shown[10] that for a single motion the maximal velocity is $\frac{30A}{16T}$, where A is the amplitude of the trajectory and T is the time of the movement. Therefore, $T_1=\left|\frac{30A_1}{16V_{max}}\right|$ can be taken as the minimal value of the time for the first movement, where V_{max} is the maximal permitted velocity.

In this model, the amplitude of the *Start-to-New* trajectory was chosen to be equal to 100 cm and the maximal permitted velocity to be equal to 187.5 cm/sec. These values are convenient since they imply $T_{i_min}=\frac{|A_i|}{100}$ and the minimal time T_{1_min} for the first trajectory is equal to 1 sec. Amplitudes of the first and second motions are denoted by A_1 and A_2, respectively.

The behavior of the chek-function was separately analyzed for three cases according to the signs of the amplitudes of the trajectories. The True values defines the search domain for $\min(S+T_2)$. This domain is termed as the *real-time search domain* or RTS domain and was evaluated for the following three cases:

1. $\mathrm{Sign}(A_{1x})=\mathrm{Sign}(A_{2x})$, $\mathrm{Sign}(A_{1y})=\mathrm{Sign}(A_{2y})$
2. $\mathrm{Sign}(A_{1x})=-\mathrm{Sign}(A_{2x})$, $\mathrm{Sign}(A_{1y})=-\mathrm{Sign}(A_{2y})$
3. $\mathrm{Sign}(A_{1x})=\mathrm{Sign}(A_{2x})$, $\mathrm{Sign}(A_{1y})=-\mathrm{Sign}(A_{2y})$

To analyze the domain's structure for each case, a special simulation program in C was written.[11] The program receives the targets coordinates and starts a loop for every

$$S:\ 0\le S\le T_1$$

and

$$t:\ T_2\le t\le 3T_2 \text{ or } 0.5T_2\le t\le T_2$$

For each loop the check-function is calculated. Then, the optimal search procedure is empirically defined for each case according to the RTS characteristics.

- Case 1: $\mathrm{Sign}(A_{1x})=\mathrm{Sign}(A_{2x})$, $\mathrm{Sign}(A_{1y})=\mathrm{Sign}(A_{2y})$

We can introduce two trivial conditions on S and T_2:

$$0\le S\le T_{1_min}$$

$$T_{2_min}\le T_2\le T_{2_min}+T_{1_min}$$

The condition $S>T_{1_min}$ corresponds to two separate motions performed sequentially and obviously does not violate the constraints. Increasing S does not improve the results.

Taking $T_2>T_{2_min}+T_{1_min}$ does not provide any improvements because we can always pick S to be equal to T_{1_min} (this is the worst case) and then the time of the final movement will be equal to $T_{2_min}+T_{1_min}$.

Figure 1a represents a model with both amplitudes (of the first and the second trajectories) greater than zero. The grey area represents the RTS domain. As can be noted from this figure the RTS domain consists of two regions where the left one becomes smaller as the ratio between the second and the first amplitude increases (A_1 is kept fixed). Such behavior corresponds to the situation when the second amplitude is relatively small and the second movement starts and ends within the first half of the first movement without violating the constraints. For larger A_2, such summation of the trajectories is impossible and this region of the RTS domain does not exist.

The minimum of the linear function $S+T_2$ must be on the boundary of the RTS domain.

For A_1 and A_2 values when a left (small) region of the RTS domain exists and for a small enough S_1 for $S=S_1$ we can pick:

$$T_2=\min_{(T_2^*,\,S_1)\in\mathrm{RTS}}(T_2^*)$$

If the two trajectories that start at the switching time S_1 cannot be summed, then the minimum will occur on the tangent line $S+T_2=C$ to the RTS domain where C is some constant. The algorithm for finding the tangent point between the line $S+T_2=C$ and the boundary of the domain will be described below.

- **Case 2: $\mathrm{Sign}(A_{1x})=-\mathrm{Sign}(A_{2x})$, $\mathrm{Sign}(A_{1y})=-\mathrm{Sign}(A_{2y})$**

A_2 is picked to be neglected, while A_1 is still fixed and positive. In this case, condition (1) for S still holds. However, for T_2 we have a slightly different inequality:

$$0<T_2<T_{2_min} \qquad (2')$$

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press

This conclusion follows from the fact that the velocity vectors for the first and second movements have opposite signs and when summed will compensate for each other. It means that we can pick the absolute value of the velocity for the second movement to be greater than the maximal permitted velocity since when the second movement is added to the first one it will reduce the total value and the final trajectory does not violate the constraints. Hence, the value of T_2 can be decreased and be smaller than T_{2_min}.

On the other hand, we can not pick T_2 to be too small because then the maximal velocity of the second movement will be too high and adding the second movement to the first one will not enable to obey the constraints. It is quite clear that the low bound for T_2 should be taken to be equal to $0.5 \cdot T_{2_min}$. The value $T_2 = 0.5 \cdot T_{2_min}$ implies $V_{2max} = 2V_{max}$ which is still compensated for by V_{1max} which has opposite direction and the resulting velocity will be less than V_{max}. However, a smaller value of T_2 results in a too high maximal velocity for the second movement and consequently in violation of the velocity constraints after the trajectory superposition.

Figure 1b represents the search domain for cases involving amplitudes of opposite signs and velocity constraints in effect. On this plot we picked the upper bound for T_2 to be slightly greater than T_{2_min} and one can see that all points such that $T_2 > T_{2_min}$ are within the RTS domain (grey strip in the upper part of the plot). One can also see the "drop-shaped" region which becomes smaller when the absolute value of A_2/A_1, increases.

The minimum of the $(S + T_2)$ expression could be found as in above on the tangent line $S + T_2 = C$ of the RTS domain.

- **Case 3: $\mathbf{Sign(A_{1x}) = Sign(A_{2x})}$, $\mathbf{Sign(A_{1y}) = -Sign(A_{2y})}$**

A two-dimensional movement could be considered as two separate one-dimensional movements combined together. Each of these movements must obey its own constraint and the minimal time of the whole movement will be defined by a stronger condition on $T_{min\,x}$ and $T_{min\,y}$. As was mentioned in the previous paragraphs, the case of amplitudes with equal signs implies $T_{2_min} \le T_2 < 3 \cdot T_{2_min}$ and the case of amplitudes with opposite signs implies $0.5 \cdot T_{2_min} \le T_2 \le T_{2_min}$. Thus, the part of the trajectory that has equal signs has also longer duration and the whole movement duration will he defined by the duration of this particular motion component.

3.1. The algorithm for finding the optimal switching time with velocity constraints

The algorithm used for finding the optimal switching time for the superposition method was performed with velocity constraints and was designed to handle the search domain represented in Figure 1. These contraints were selected for implementation since they are the most common in industrial robot control.

The first step of the algorithm is to define the signs of trajectories and to decide which type of search domain to consider. The algorithm generates the optimal S and T_2, but the whole time of the movement is defined as $\max(S + T_2, T_1)$. It means that the motion time cannot be less than T_1 even if S and T_2 are small. If for some particular case we get $S + T_2 \le T_1$ it makes sense to increase the time T_2 for the second trajectory and finish the second movement at the same time as the first one. This approach allows to perform the second movement smoothly without increasing the final time for the whole movement. Thus, if as a result of computation $S + T_2 \le T_1$, then T_2 is set to be equal $T_1 - S$.

Finding the optimal switching time for equal sign amplitude. The first step of this procedure is to check whether the second trajectory can be performed within the first half of the first trajectory. This corresponds to the case where the left small part of the RTS domain exists (region A in Figure 2) and S_1 is small enough. The algorithm tries to find the switching point S and duration T_2 in region A if the following two conditions hold: $S_1 < 0.25 \cdot T_1$ and $A_2/A_1 < 0.4$. If one of these conditions fails, then either region A

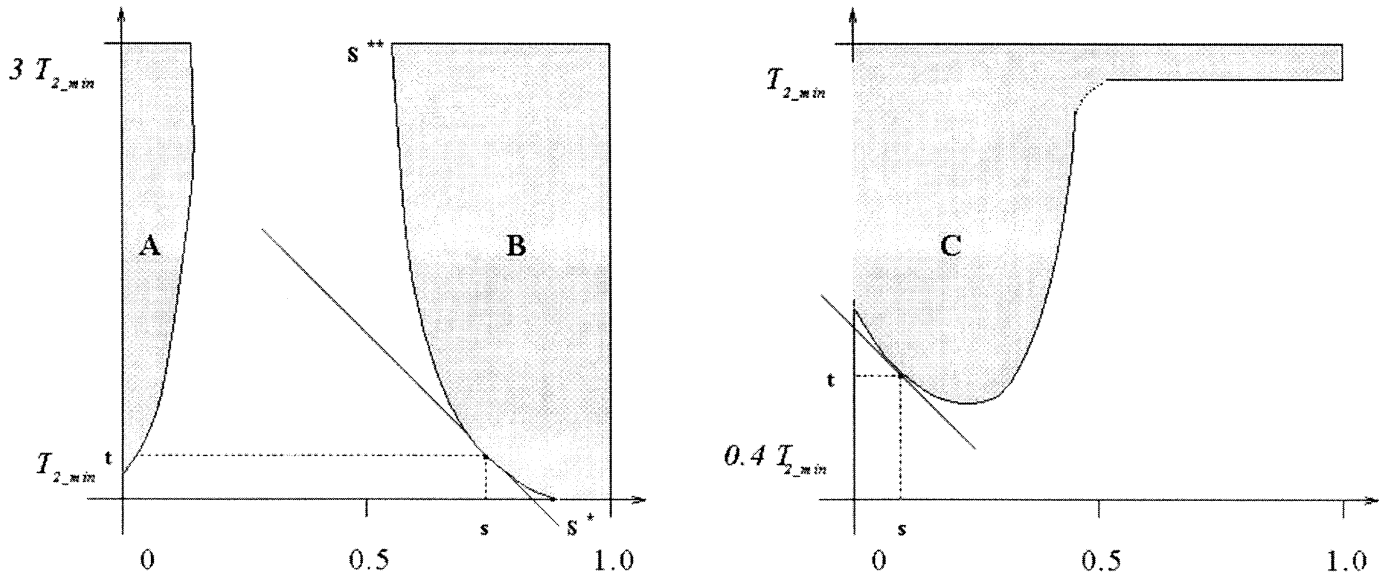


Fig. 2. Finding the optimal solution using the RTS domain.

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press

does not exist or S_1 is too large and region A is useless for us. If both conditions hold, the algorithm runs a simple numerical procedure to find the point $(S, T_2) \in$ region A. The algorithm initiates S to be equal to S_1 and tries to find the first possible value T_2.

If the optimal point T_2 is found using the above algorithm we can superimpose immediately the second trajectory ($S = S_1$) into the first one and finish the whole procedure.

If under the given conditions region A of the domain does not exist or S_1 is too large then the algorithm starts processing part B of the domain. First, point S* is found. Obviously for $s > S^*$ we can superimpose the second trajectory with $T = T_{2_min}$ and we do not violate the constraints. S* is a right bound for the domain within which we actually search for the ophmal S and T_2. It means that only $s \le S^*$ should be considered.

The right bound S* is found using a fast binary search algorithm. The program considers the search region between S_1 and T_1, divides it into two halves and depending on the value returned by *Check-function* chooses one of the halves of the region as a new search region for the next iteration.

After the point S* is found the algorithm progresses along the boundary of the domain and calculates the sum of s ($s \le S^*$) and t corresponding to it (Figure 2). The boundary value t is calculated using the same binary search as described above. That is, for every s the binary search algorithm searches in the region from T_{2_min} to $3 \cdot T_{2_min}$ and produces t corresponding to the given s.

The sum of s and t monotonically decreases down to its minimal value. The values of s and t which give the minimal sum are the desired optimal parameters of the trajectory superposition. When the sum of these values bcgins to increase while thc algorithm goes from S* to S_1 the algorithm is stopped and returns S equal to the current s and T_2 equal to the current t as the optimal switching time and duration for the second trajectory. This criterion can be used since the function s + t for s and t corresponding to the boundary has only a single minimum.

Finding the optimal switching time for opposite sign amplitudes. If two amplitudes of the two motions have opposite signs the RTS domain has the “drop-shape” shown on the right domain of Figure 2 (region C). A domain of this form is handled by another procedure which consitutes the second part of the algorithm.

As mentioned above, the optimal time for the second movement can be made even less than T_{2_min}. This can occur if the second movement starts within the first half of the first movement and its amplitude is relatively small. The algorithm for finding the optimal switching time in this case also uses the fast binary search algorithm. Since the minimal possible switching time is S_1, the algorithm starts processing from $s = S_1$. For every $s \ge S_1$ the corresponding t is calculated. The point (s, t) is located exactly on the boundary of the domain and is calculated in the same way as in the previous procedure. The sum of s and t is calculated and the algorithm terminates using the same criteria: the minimum is found when the sum stops decreasing.

3.2. Complexity of the real-time superposition algorithm

The time to find an optimal switching time is strictly dependent on the control interval Δt. The RTS domain is defined on the set $(0, T_{t_min}) \times (T_{2_min}, 3 \cdot T_{2_min})$. Suppose our space has dimension $n \times 2m$, that is we have n intervals in $(0, T_{1_min})$ and m intervals in $(0, T_{2_min})$. Suppose also that the complexity of the *Check-function* is N*. The complexity of each branch of the algorithm must be determined separately.

- **Equal signs of amplitudes:** In region A the complexity of the search can be estimated as $C_A = 0(mN^*)$. The processing of region B first takes O(log nN*) to find the point S*. If the number of points between right bound S* and the actual minimum is K_b then the complexity of finding the minimum is $O(\log mK_bN^*)$. Therefore, the whole complexity can be estimated as:

$$C_{A-B} = 0(mN^* + \log nN^* + \log mK_bN^*)$$

- **Opposite signs of amplitudes:** Let K_c be the number of points from left bound S_1 to the actual minimum S. The complexity can be estimated as

$$C_C = O(K_c \log mN^*)$$

The algorithm executes either region A–B or C with equal probability. Hence, the overall complexity can be defined as the maximum of complexities A–B and C.

The complexity of the Check-function N* can be estimated as O(m) (the complexity of search within each bell-shaped trajectory is O(m) and so is the summation including checking of the constants during the summation).

Hence, the final estimation of the algorithm complexity is as follows:

$$C_{alg} = O(m^2 + m \log n + nm \log m)$$

where K_b is estimated as O(n).

4. IMPLEMENTATION OF THE SUPERPOSITION ALGORITHM

The superposition algorithm was implemented and tested on an industrial Adept-one robotic system. A diagram of the physical set-up is shown in Figure 3.

4.1. The Adept-One robot

The Adept-One manipulator is a four degrees of freedom SCARA direct drive arm. Only the first and second axis were controlled providing motion in the horizontal plane.

The language used for programming the Adept-Controller is *VAL-II*. This language is interpreted, not compiled into robot commands and therefore is extremely slow. Such an interface permits only position and partial velocity control. The controller receives Cartesian or joint space position commands. Only Cartesian coordinates were used in the

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press

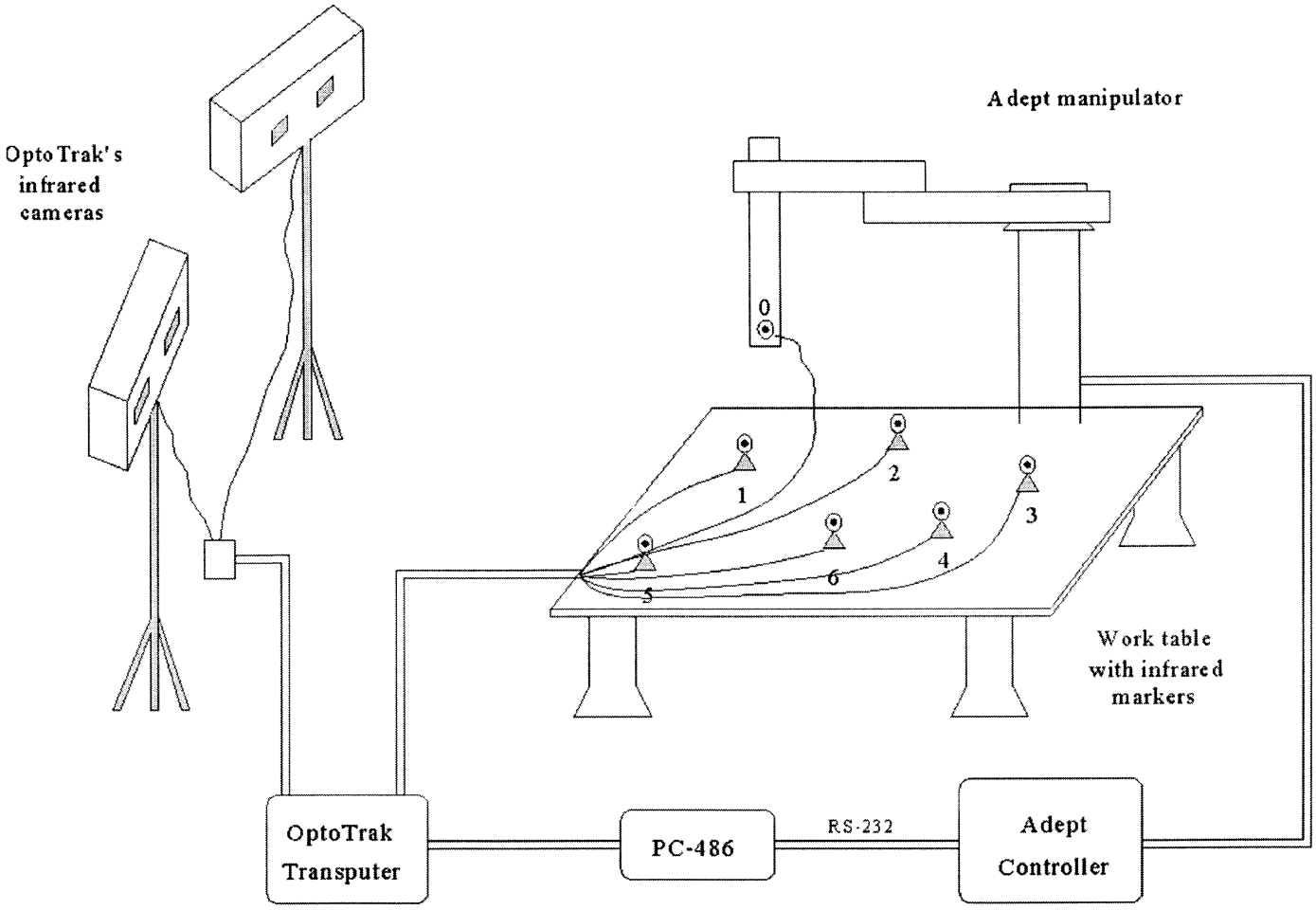


Fig. 3. A diagram of the experimental setup.

experiments performed. The software written in *VAL-II* is very slow and therefore was not suitable for real-time implementation of the superposition trajectory control scheme. Hence, all position computations were performed on an external PC-486 (33 Mhz) computer. The output of the external control program (the actual position to move to) was sent through a serial interface to the Adept controller for execution.

4.2. The OptoTrak system

An OptoTrak system[12] vas used to determine the target's 3-D position. As targets we used infrared emitting diodes that served also as markers for the OptoTrak system. The infrared light emitted by rhe marker is detected by two infrared sensitive cameras. The 3-D position is transmitted to the PC. The real-time accuracy of the calculated position was using dedicated software developed was 0.1 mm.

4.3. Description of experiments

Six markers were placed on the Adept's work table. These markers represented six different targets for the movement control scheme. One additional marker was placed on the end-effector of the robot and was used to obtain the actual position of the end-effector during its motion.

The visual targets were detected using the OptoTrak infrared system. First, the system is initialized. This process involves downloading control sottware into the OptoTrak transputer and the Adept-controller. Then the program sends the robot arm to the initial position which is chosen to be at the location of marker number 1. The program queries the OptoTrak system about the location of the first marker and commands the manipulator to move to this location. When the motion terminates, the program waits for the user's input which should be the number of the next marker to move to. When the system is notified of the second target, the program obtains its coordinates from the OptoTrak system, generates a minimal jerk trajectory from the initial target position to the second target, implements the superposition scheme and moves the robot arm according to the generated trajectory.

During the motion of the robot arm, the program waits for one more user's input, which should be the number of the new target. Once the system is notified of the new target location, the program starts the superposition algorithm and updates the current trajectory according to this scheme. Depending on the calculations, the robot arm can either proceed with the current trajectory all the way to the calculated switching point or switch to the new trajectory immediately. During the motion, the program is ready for an

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press

input of a new target. If this input is received, the number input to the system becomes the number of the new target (marker) and the program moves the robot arm to this target calculating the new trajectory according to the superposition algorithm in real-time.

5. RESULTS

The performance of the algorithm is strictly dependent on the number of search points we divide our amplitudes into (m and n in the complexity formula). Let us define these parameters for our particular experiments.

In the current setup, the maximal distance between markers (targets) was set to be about 100 cm. The maximal permitted velocity for both the X and Y axes was chosen to be 10 cm/sec. Thus, the maximal time for every target-to-target motion is $\frac{30 \cdot 100}{16 \cdot 10} \approx 18.7$ sec. The time of the minimal movement segment Δt was taken to be 0.13 sec. This implies that the maximal number of points n is equal to 144. This is the number of points along the S axis ($0 \leq S \leq T_1$). The step along the T_2 axis was chosen to be the same and respectively the number of points m was equal to 144.

To generate a movement between two points the controller needs 0.032 sec. This means that the superposition algorithm has approximately 0.1 sec to find the optimal switching time and re-plan the current trajectory. For the experiments conducted, 0.1 sec turned out to be the minimal time required for the program run on a PC-486, 33 MHz to finish the calculations and modify the trajectory before the next point will be requested by the controller. This is the necessary condition to obtain a continuous movement.

Decreasing Δt causes an increase in the number of samples n and m and thus an increase in the computation time. On the other hand, the time which is required for the algorithm to complete the computations is defined as $\Delta t - 0.032$. Thus, if we decrease the value of Δt when a new target appears and the superposition algorithm is started, it can not find the optimal switching time within $\Delta t - 0.032$ sec and the controller must wait for the next sequential trajectory point while the previous motion segment already is completed. As a result of this computational delay, the robot stops and the current motion is interrupted.

5.1. One-switch movements

Figure 4a shows the trajectory of the arm when only one switch of the target occurred. The Start (initial) target is at marker no. 1, the End target is at marker no. 6 and the New target which the robot might switch to is marker no. 4. The information about the New target was supplied to the program when the robot has covered about half of the distance between the first and sixth markers. The trajectory modification occurred however, only at point B which is about $\frac{3}{4}$ of the distance away from the Start target. Thus, within the A–B motion segment only the first movement was performed. At point B the superposition began and during segment B–C both motions were active. During the motion along this segment, the robot changed the direction of the movement as smoothly as possible. The first movement was completed when the second one was still in progress (point C) and as a result the last segment C–D of the trajectory was a straight line which corresponded to the second movement.

The velocity profile (Figure 4b) was derived from two "bell-shaped" profiles summed together starting at the switching point (point B). This is the actual velocity profile obtained by measuring at the end of every control interval Δt the position of the marker number 0 placed on the robot arm.

Since the velocity of the robot arm cannot be controlled during motion between two end points, it was approximated by small motion segments with a constant velocity. Thus, the actual velocity profile was found to be less smooth than specified by the mathematical model. This was also caused due to errors in the OptoTrak measurements.

5.2. Multi-switch movements

A multi-switch movement is represented in Figure 5a (trajectory) and Figure 5b (velocity). The starting target is at marker no. 1 and the sequence of the new targets is 2, 4, 6 and 2. It is assumed that the knowledge about the new targets is incremental. That is, when the information about the new target arrives we do not change any of the previous units, but just compute T_i and S for the new trajectory and add it on top of the existing ones watching out not to violate any constraint.

One can see that if the switch is made at the end of the trajectory (for example near marker four), when the speed of the arm is relatively small, the radius of the turn is also small. That is, the low velocity enables a sharper turn without increasing the jerk. On the other hand, if the switch occurs at the middle of the trajectory, when the speed is close to its maximum (for example on the segment located between marker four and six) then the turn generated by the algorithm has a larger radius and the change in the movement direction is performed more gently.

Concerning the movement velocity profile, the local minimum of velocity marked at A corresponds to the sharp turn near marker four, while the local minimum marked at B corresponds to the smooth trajectory turn near marker six.

6. CONCLUSIONS

A real-time algorithm for trajectory modification has been developed and implemented based on the superposition scheme. The algorithm generates optimal re-plan parameters (the switching time and the duration of the second trajectory) using a special search domain fixed for each type of constraints. The search and generation of the new trajectory are performed within the robot motion constraints. The algorithm was tested on an industrial robotic system and its feasibility was demonstrated. Several experiments were conducted to study the behavior of the algorithm for one and multi-switch movements. The

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press

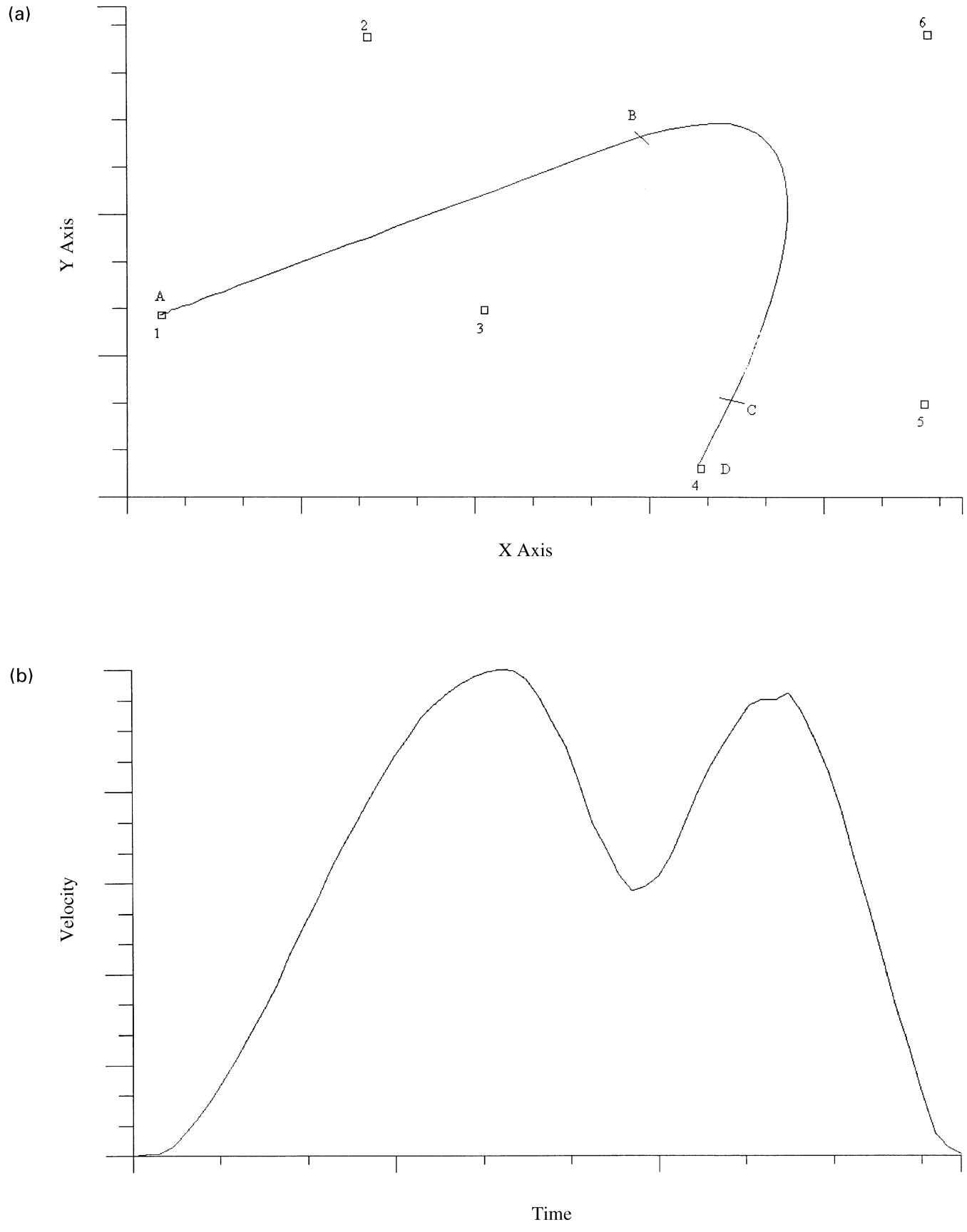


Fig. 4. A one-switch movement: (top) trajectory results; (bottom) velocity profle.

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press

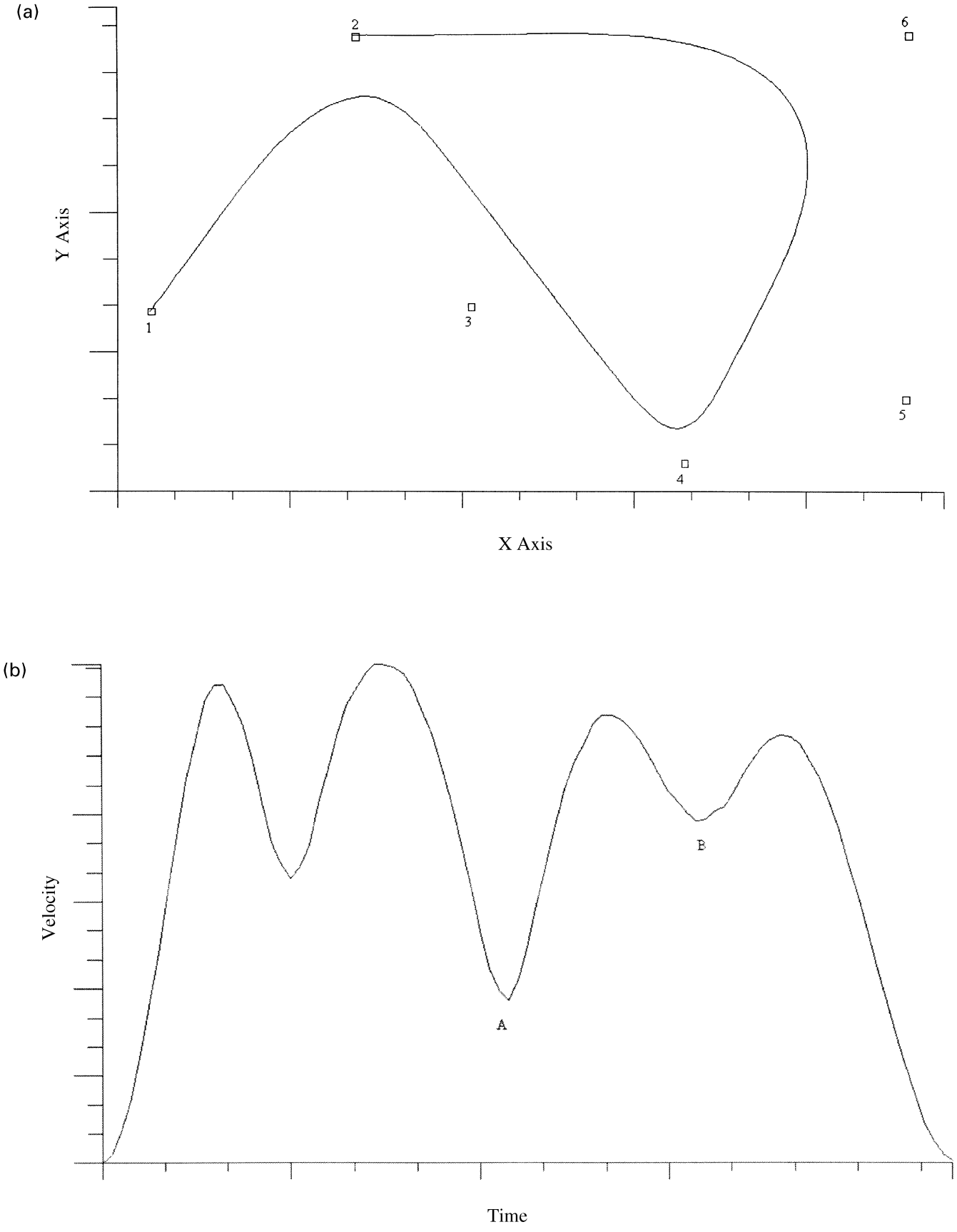


Fig. 5. A multi-switch movement: (top) trajectory results; (bottom) velocity profile.

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press

algorithm was implemented only for velocity constraints because these are most typical and natural types of constraints for industrial robots. Including other types of constraints and their different combinations should be the subject of fuhure research.

To obtain a continuous movement, the calculations of the modified trajectory must be performed within the controller time constraints. This constrains the number of points searched. Otherwise, the current motion is stopped due to the computational delays.

REFERENCES

1. J. Lloyd and V. Hayward, "Trajectory generation for sensor-driven and time-varying tasks", *Int. J. Robotics Research* **12**(4), 380–393 (1993).
2. T. Park and B. Lee, "An approach to robot motion analysis and planning for conveyor Tracking", *IEEE International Conference on Robotics and Automation* (1991) Vol. **1**(3), pp. 70–75.
3. R. Sharma, "Locally efficient path planning in an uncertain, dynamic environment using a probabilistic model", *IEEE Transactions on Robotics and Automation* **8**, 105–110 (1992).
4. J. Carusone, K. S., Buchan and O. M. T. D'Eleutesirio, "Experiment in end-effector tracking control for structurally flexible space manipulator", *IEEE Transactions on Robotics and Automation* **9**(5), 553–570 (1993).
5. T. Gat-Fallik and T. Flash, "The superposition strategy for arm trajectory modification in robotic manipulators", *IEEE Transactions on Systems, Man and Cybernetics* **Part B 29**(2), 93–99 (1999).
6. K. Takayama and H. Kano, "A new approach to synthesizing free motions of robotic manipulators based on a concept of unit motions", *IEEE Transactions on Systems, Man, and Cybernetics* **25**, 453–463 (1995).
7. E. Burdet, "Algorithms of human motor control and their implementation in robotics", *Ph.D thesis* (Institute of Robotics, ETH, Zurich, Switerland 1996).
8. T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model", *J. Neuroscience* **5**(7), 1688–1703 (1985).
9. T. Flash and E. Hennis, "Arm trajectory modification during reaching towards visual targets", *J. Cognitive Neuroscience* **3**(3), 219-230 (1991).
10. T. Gat, "The superposition strategy tor trajectory modification: extensions to robot dynamic tasks", *M.Sc. thesis* (Weizmann Institute of Science, Rehovot, Israel, 1990).
11. V. Rogozin, "Robotic control for dynamic tasks", *M.Sc. thesis* (Dept. of Applied Mathematics and Computer Sciences, Weizmann Institute of Science, Rehovot, Israel, 1993).
12. Optotrak, "Optotrak technical product description", *Optotrak, 3D motion measurement system* (Northern Digital Inc., 403 Albert St., Waterloo, Ontario, Canada, N2L 3V2, 1991).

https://doi.org/10.1017/S0263574701003393 Published online by Cambridge University Press