Accelerated Needle Steering Using Partitioned Value Iteration

Ali Asadian, Mehrdad R. Kermani, and Rajni V. Patel

Abstract—This paper presents a fast 2D motion planner for steering flexible needles inside relatively rigid tissue. This approach exploits a nonholonomic system approach, which models tissue-needle interaction, and formulates the problem as a Markov Decision Process that is solvable using infinite horizon Dynamic Programming. Starting from any initial condition defined in the workspace, this method calculates a set of control actions that enables the needle to reach the target and avoid collisions with obstacles. Unlike conventional solvers, e.g. the value iterator, which suffers from the curse of dimensionality, partitioned-based solvers show promising numerical performance. Given a segmented image of a workspace including the locations of the obstacles, the target and the entry point, the partitioned-based solver provides a descent solution where high resolution is required. It is shown in this paper how prioritized partitioning increases computational performance of the current DP-based solutions for the purpose of off-line path planning. By default, our planner selects the path with the least number of turning points while maintaining minimum insertion length, which leads to the least damage to tissue. In this paper, more emphasis is given to the control aspects of the problem rather than the corresponding biomedical issues.

I. INTRODUCTION

Medical intervention using steerable needles has become a common minimally invasive procedure in recent years. Facilitating curved trajectories, flexible needles could be utilized for localized drug delivery or tissue biopsy especially in dealing with regions that are difficult to access. Therefore, robotics-assisted needle steering, which is intended to guide the needle to specific targets inside soft tissue, has turned into an active research area. To this end, a robot can be used to control the base of a flexible bevel-tip needle. A desired trajectory for the needle tip is then planned such that the needle does not penetrate delicate structures such as nerves and blood vessels or collide with bones. In this respect, targeting inaccuracy stems from a variety of factors such as (1) organ deformation (2) needle bending mechanics as a result of tissue-needle interaction (3) inhomogeneity, nonlinear viscoelasticity, and anisotropy of real organic tissue (4) target movement due to respiration, heartbeat or similar artifacts, and (5) tiny anatomic structures which could not be easily identified with a common imaging modality.

Depending on the application, the desired placement accuracy and the way by which a flexible tool is guided varies. Several researchers have struggled to simplify the problem and to find a reasonable solution through numerical models, though such models can never perfectly mimic tissue complexities [1]. Thus, open-loop execution of model-based trajectory planners may not be feasible in most cases, due to the sensitivity to deviations in tissue-needle characteristics.

To avoid black-box modeling and to explore a closed-loop control rule, a thorough study of biophysical aspects of the problem has to be done in advance, i.e., to steer a needle in any percutaneous procedure, extracting a model for the deflection and the associated curvature is the first step.

With regard to modeling, Okamura and Simone [2] developed an empirical model for bilateral tissue-needle insertion force. According to their model, the stiffness force occurs before the puncture of the capsule while the friction and cutting forces occur right after the main puncture. They also characterized the impact of the needle diameter and the tip type using a silicone rubber phantom. Podder et al. [3] derived a statistical model to estimate the maximum force that the needle experienced during insertion into the prostate and the perineum. Force data were collected during prostate brachytherapy to build the model based on (1) patient-specific parameters including prostate volume, and (2) procedure specific criteria such as needle size.

Most of the biophysics-based models, which incorporate monitoring of elastic medium properties, might not be amenable to real-time control or be applicable to in vivo tests even if a fast Finite Element Model (FEM)-based method is used. Generally speaking, FEM is accurate for modeling small linear elastic deformations but its computational burden is very high and the obtained accuracy is much dependent on its inputs. On the other hand, mass-spring-damper models may work fine for real-time simulations but they have limited accuracy [1]. Mechanics-based approaches consider only the kinematics of the needle by analyzing the geometry of the needle and the measured forces at the base. Abdolhassani and Patel [4] modeled a beveled tip needle as a cantilever beam and estimated the amount of deflection using Euler-Bernoulli beam theory. Misra et al. [5] explored the sensitivity of tip axial and transversal forces to tissue rupture toughness, linear and nonlinear tissue elasticity as well as bevel angle of the needle. Webster et al. [6] developed a "bicycle" model experimentally for steering a flexible beveled tip needle in relatively stiff materials, and described this motion as a nonholonomic kinematic system. Taking advantage of this model, needle bending has been studied by many researchers where steerability is provided by asymmetric forces acting at the needle tip. Alterovitz et al. [7], [8] formulated planar needle steering as a Markov Decision Process (MDP) and applied Dynamic Programming (DP) to find a simulation-
based optimal planner, which computes the path that is most likely to succeed. As a basis for the current study, this method is reviewed in the next section.

Other methods of steering have been addressed in the literature with more emphasis on the biophysical side of the problem rather than the control strategy. Incorporating soft tissue motion, needle flexibility and a physically based contact model, DiMaio and Salcudean [9] introduced a manipulation Jacobian for 2D steering based on potential fields technique. Note that even in the simplest experimental case with no external artifacts, the needle motion is governed by a non-linear dynamics. Thus, non-linear control or non-linear optimization is a remedy to steer the needle.

Clearly, a more accurate solution requires a more complex interaction model leading to a less feasible planner. In order to obtain an efficient steering method in the context of control theory, our work builds on the novel method presented by Alterovitz et al. [7], [8]. The bicycle-like model is formalized as an MDP, and the path planning task as the corresponding DP solution. Using common numerical solvers, e.g. the value iteration, the curse of dimensionality poses a major hurdle, and this issue is exacerbated especially where high resolution is employed. The performance of the value iteration can be effectively improved by removing redundant states or by updating them in a proper order. In the conventional value iteration, updating states close to the start point, based on a simple inference, might not be computationally optimal since the corresponding successors are far from the correct values. On the other hand, updating states in the vicinity of the goal would not help either because their values have converged properly. Thus, some regions between the start and goal points are supposed to have the maximum productivity in terms of the required number of updates for a single solution. This idea is effectively realized in the current work. Briefly, instead of solving DP over the entire state space, which is computationally expensive, it is solved within a specified partition and then uses a flow metric to jump to the next partition. We will show how steering performed within a partitioned space can be more efficient than that in a non-partitioned version. This computational advantage, which is particularly important in the context of real-time planning, comes at the cost of some extra memory and overhead to represent the partitioned system. The proposed method suggests a non-linear optimal solution for the needle steering in the presence of stochastic deflection.

This paper is organized as follows. Section II briefly reviews the bicycle needle steering model used in the current work. Section III describes the generic MDP, the conventional value iteration solver and its counterpart in partitioned state space. In section IV, simulation results are given to compare the efficiency of the new version, and section V presents conclusions and future work.

II. NEEDLE STEERING MODEL

To explore a model-based approach for guiding the needle, first we need to model tissue-needle interaction. In both the rigid and flexible cases described in the literature, the inputs at the base of the needle can be represented as inputs to a kinematic system. To achieve this, we obtain the desire control action using the bicycle model developed by Webster et al. [6]. Note that choosing a rigid material to represent tissue may not be realistic as was done in [6]; however, it is acceptable for preliminary experiments and to allow study the effects of the needle deflection without the complexities arising from tissue behavior.

It is presumed in this work that the tip position is measured by an imaging modality such as X-ray fluoroscopy, and the workspace and obstacles are extracted by segmenting the 2D cross-section of the image. Translational and rotational velocities are constants in our model, and partial observation is beyond the scope of this paper.

One DOF actuated at the needle base is a pure insertion of length $\delta$ while the second DOF is a pure tip rotation of $180^\circ$. Under this condition and regardless of the insertion velocity, the needle moves along an arc of approximately constant radius $\rho$ in the direction of the bevel. A $180^\circ$ rotation of the needle causes the bevel at the tip to point in the opposite direction of the preceding arc of deflection. Thus, the entire motion of the needle is fully described by the motion of the needle tip and a number of circular segments with the radius $\rho$. The value of $\rho$ is dependent on the tissue-needle interaction properties, which includes tissue elasticity, needle geometry, friction, and clamping forces during insertion. If one can determine the aforementioned turning points inside the tissue, the planning problem becomes straightforward since the rest of the actions are merely pure insertions.

Note that control action is applied at discrete time intervals. Thus, investigating a proper strategy for a discrete representation of the system kinematics is a major part of the modeling. In this regard, the scheme proposed in [7], [8] is outlined next. At any points of workspace, a binary control action is applied. One control action is a pure insertion of length $\delta$ and the other one is $180^\circ$ rotation followed by an insertion of the same length. The former action is represented by $u_k = -1$ while the latter is represented by $u_k = 1$.

Tracking circular segments, the needle state at instant $k$ is defined as $S_k = (x_k y_k \theta_k b_k)^T$ where the tip position $t_k = (x_k y_k)^T$ and the tip orientation angle $\theta_k$ are rounded values to the nearest points on corresponding projection planes. For this approximation, we overlay a control circle with radius $\rho$ on a $\Delta$-grid plane consisting of horizontal and vertical $\Delta$ spacing. Each control circle is also divided into $N_r$ discrete arcs of the same length of $\delta$ where,

$$\delta = \frac{2\pi \rho}{N_r} = \rho \alpha$$

In this representation, $\Delta$ and $\alpha$ are spatial and angular resolutions, respectively. At each position on the $\Delta$-grid network, the needle may be in any of the $N_r$ orientation states and in any of the two clockwise or counter-clockwise directions. Each direction corresponds to two control circles that are tangent to each other at $t_k$. The bevel direction or $b_k$ is a binary variable which keeps the history of the previous arc and switches at turning points or when $u_k = 1$. 

2786
The tip angle or $\theta_k$ is the tangent angle of the corresponding control circle. From an initial value, the value of $\theta_k$ increments or decrements by $\alpha$ depending on the instantaneous control action and the current circular segment. Fig. 1 demonstrates the projected needle motion on the imaging plane through segmented arcs. Consequently, this selection results in a model with $n$ discrete states given by,

$$n = \frac{4\pi \rho x_{\text{max}} y_{\text{max}}}{\Delta^2 \delta}$$

where $x_{\text{max}}$ and $y_{\text{max}}$ are the depth and the height of the imaging plane. Accordingly, a better resolution implies a larger number of discrete states.

As mentioned earlier, the tip may be deflected from its intended path as a result of unpredictable tissue-needle interaction, local tissue displacements or any other artifacts. To describe this complex behavior, $\theta_k$ and the value of $\rho$ have to be updated intraoperatively. Since the value of $\rho$ seems to be more deterministic in this case, it is set to a fixed value. Finally, deflection from circular segments was modeled using a Gaussian distribution of a sequence of five discrete values as defined in (1). In this definition, $\beta_i (1 \leq i \leq 3)$ specifies statistical properties of the deflection as,

$$P\{\beta_k = m\alpha\} = \begin{cases} 
\beta_1 & \text{if } m = 0 \\
\beta_2 & \text{if } m = \pm 1 \\
\beta_3 = \frac{1}{2}(1 - p_1 - 2p_2) & \text{if } m = \pm 2 
\end{cases}$$

At the end of each iteration, correction is made by simply adding $\beta_k$ to $\theta_k$. Here, $\beta_k = 0$ corresponds to the deterministic motion with no deviation.

### III. Dynamic Programming and Partitioning-based Solution

As shown by Alterovitz et al. [7], [8], MDP and DP can be employed to yield an optimal path planner for the needle steering problem. Since the current study applies a partitioned version of this algorithm, a brief introduction to generic MDP and DP is in order [10].

MDP is a four-tuple $(S, U, T, C)$, where $S$ is the set of states that describes the system at a given time. At any stage $k$, each state has an associated set of applicable actions $U_k$, and the Markov property requires that the effects of an action taken at one state depend only on that state and not on the prior history. The transition function for each action, $T : S \times U \times S \to [0, 1]$, specifies the probability of changing to state $s$ after applying $u$ in state $s$, and $C : S \times U \times S \to R$ determines the immediate transition cost. Given an MDP, we define a policy or $\pi : S \to U$ as a mapping from states to actions. An optimal policy determines how actions at different states are chosen in order to minimize the expected cost. Value function or $V, V : S \to R$, gives the minimum value of the total expected cost from being in a state $s$ as,

$$V_\pi(s) = E \left[ \sum_{k=0}^{\infty} \gamma^k C(s^k, u_k, s^{k+1}) | s_0 = s \right]$$

where $E$ is the expected value operator and $\pi \in U = \{u_0, u_1, \ldots\}$ is the set of applied control actions ($0 \leq \gamma \leq 1$).

The Bellman Optimality Equation shows that the expected value of a policy can be computed through,

$$V^*(s) = \min_{u \in U(s)} \sum_{s' \in S} \left[ C(s, u, s') T(s, u, s') + \gamma T(s, u, s') V^*(s') \right]$$

where,

$$V^*(s) = \min_{\pi} V_\pi(s)$$

The Bellman residual of a state $s$ is defined to be the difference between the value functions in consecutive iterations, and the Bellman error equals the maximum residual of the states. When this value is less than a threshold $\lambda$, it is concluded that the value functions have converged sufficiently. Consequently, an optimal policy is obtained by,

$$\pi^*(s) = \arg \min_u \sum_{s'} \left[ C(s, u, s') T(s, u, s') + \gamma T(s, u, s') V^*(s') \right]$$

Value iteration is a numerical solver that sweeps through the state set of the MDP and updates the value functions. Iteratively, this process allows us to start with any estimate of the value function and converge to the true value function,

$$V_k(s) = \min_u \sum_{s'} \left[ C(s, u, s') T(s, u, s') + \gamma T(s, u, s') V_{k-1}(s') \right]$$

In the case of a flat MDP, with $n$ states and a maximum of $m$ admissible actions for any state, the standard value iteration requires at most $O(mn)$ and $O(mn^2)$ operations in deterministic and stochastic cases, respectively. In light of this fact, the convergence rate becomes very low when a high resolution is considered in motion planning. The main drawback of this method is that at each iteration, the value function of every state is updated, even if such an update cannot contribute to the convergence. Roughly speaking, if the values of the successor states are incorrect, updating $V^*(s)$ will not move the value function any closer to $V^*(s)$.

Several approaches have been proposed for solving large MDPs, and thus for proper allocation of CPU resources. Hierarchical dynamic programming and clustering using geometric graphs and topological maps is a remedy to increase efficiency in this case [11]. The use of subtasks
to compress the scales of the state space and to identify maximally productive regions is another way proposed by Wingate and Seppi [12]. They upgraded their initial work by presenting a prioritized version of the partitioned model to yield an efficient computation and to avoid redundant updates in a deterministic mode [13], [14]. The current study takes advantage of this method so as to circumvent the computational problem in large-scale environments.

The idea is to correct the estimated value functions for the current state and then for all dependant states. Similar to the traditional version, the Bellman error characterizes the usefulness of any given update, while metrics constructed based on the Bellman error are employed as priority measures in a priority queue. Assuming that $P$ denotes a set of partitions, which tessellates $S$, the partition-based method can be described using the definitions and notations as listed in Table I [14]. The Potential Information Flow (PIF) is a numerical metric which determines the amount of information that could flow when an update is executed.

$$B_k(s) = \max_u \sum_{\hat{s}} [C(s, u, \hat{s})T(s, u, \hat{s}) \gamma \sum_{v} \min_{\delta} T(s, u, v) - V_k(s)]$$

(7)

This selection is equivalent to the first metric $H_{1k}(s)$ introduced in [14], and represents the potential change that an update could lead to. Note that this prediction is not the same as the actual difference between the value functions of two consecutive samples. For problems with a limited amount of time to be solved, setting PIF as $H_{1k}(s)$ is a good choice that changes the value function estimation by the largest possible amount toward its optimal value. Another selection as the second prioritization metric or $H_{2k}(s)$ is defined as,

$$H_{2k}(s) = \begin{cases} B_k(s) + V_k(s) & \text{if } V_k(s) > 0 \\ 0 & \text{if } V_k(s) \leq 0 \end{cases}$$

(8)

Compared with $H_{1k}(s)$, $H_{2k}(s)$ exhibits a different behavior and outperforms in some examples. The question as to which metric should be employed naturally arises, but it is not straightforward to find topological features which accurately mimic the performance of each metric. Using $H_{1k}$ or $H_{2k}$ as $H_k$, priority between two partitions and priority of a single partition are respectively defined as,

$$HPP_k(p, \hat{p}) = \max_{s \in \{p\} \cap SDP(\hat{p})} H_k(s)$$

(9)

$$HP_k(p) = \max_{\hat{p} \in P} HPP_k(p, \hat{p})$$

(10)

Finally, the pseudocode of prioritized-partitioned value iteration (PPVI) is as follows.

Initialization:

(1) $k = 0$

(2) $\forall s \in S :$

- $V_0(s) = 0$
- $H_0(s) = \min_{u \in U(s)} \sum_{s' \in S} C(s, u, s') T(s, u, s')$

(3) $\forall p, p' \in P (p \neq p') :$

- $H_{P0}(p) = \max_{s \in p} H_0(s)$
- $H_{P0}(p, p') = 0$

(4) $p_0 = \arg\min_{p \in P} H_{P0}(\hat{p})$.  

Main loop of PPVI:

(1) Solve regular value iteration and update value functions over $p_k$ until convergence.

(2) $\forall \hat{p} \in SDP(p_k)$, update partition priorities for all dependant partitions as below.

- $h_k = 0$
- $\forall s \in \{SDP(p_k) \cap \hat{p}\} :$
  - Update $H_k(s)$ using the flow metric
  - $h_k = \max(h_k, H_k(s))$
- $HPP_k(\hat{p}, p_k) = h_k$
- $HP_k(\hat{p}) = \max_{\alpha \in \hat{p}} HPP_k(\hat{p}, \alpha)$

(3) Select next partition with the highest priority as $p_k = \arg\max_{p \in P} HP_k(\alpha)$.

(4) $k \leftarrow k + 1$

(5) Repeat the main loop until one of stopping criteria becomes satisfied, i.e. $\|B_k(s)\|_{\infty} < (1 - \gamma)\lambda$.

Inside each main loop of PPVI, there is another iterative loop of a regular value iterator, which is realized by (3) to (5). This loop solves the Bellman equation on a single partition that is computationally less expensive. Once $\max(\Delta V_{k,p}) - \min(\Delta \bar{V}_{k,p})$ drops below $\lambda$, the convergence of the inner loop is satisfied. $V_{k,p}$ is defined as the value function over members of $p_k$. Thereafter, the flow metric and the priorities of local partitions are updated and the algorithm transitions to the next candidate partition with the highest priority selected from the local queue. Note that this method relies on the assumption that the corresponding MDP is positively bounded, i.e., $\sum_{s} C(s, u, s') T(s, u, s') > 0$. For this to be realized, scaling of the cost function, which does not change the optimal solution, is a good option to be performed.

In the current study, partitioning was performed by overlaying a regular grid on the previously described $\Delta$-grid network. Using this simple scheme, it is easy to evaluate the impact of partition size on the planner’s performance. Note that prior to the simulation, all subsets introduced in Table I should be computed and saved in the memory.

IV. IMPLEMENTATION DETAILS AND SIMULATION RESULTS

In our experiments, the workspace is defined as a 2D rectangle with dimensions $x_{max} = 12$ and $y_{max} = 8$. Obstacles are represented by nonconvex polygons, and the target region is defined by a circle with radius 0.2 located at point $t_{goal} = (10, 5)^T$. All dimensions are in centimeter.
We assume that $\rho = 4$, $\Delta = 0.1$, and $N_r = 60$ leading to $n = 1,176,120$ as the number of states. For simplicity as addressed earlier, grids of $n_x \times n_y$ represent partitions where $n_x$ and $n_y$ are the number of equally spaced lines along $x$ and $y$ axes, respectively. By some trial and error, we set $\gamma = 0.95$ and $\lambda = 0.01$. Noise parameters are also selected as $\beta_1 = 0.466$, $\beta_2 = 0.24$, and $\beta_3 = 0.027$ resulting in $\mu_{\beta_1 \alpha} = 0$ and $\sigma_{\beta_1 \alpha} = 0.85$. The planner was evaluated using MATLAB on a desktop computer with 512MB RAM and a single core 2.4GHz Pentium IV 32 Bit processor.

Here, failure is defined as colliding with an obstacle, exiting the feasible workspace, or reaching a state in which one of the two former incidents is negligible at the next state. Target states transition to themselves with probability 1 and zero cost regardless of $u_k$, and obstacle states transition to a termination state with probability 1. Transition probabilities for the rest of the states are determined by the non-linear kinematics discussed in section II. Two matrices of $T(s, \pm 1, s')$ appear in the solver, each of which contains $n^2$ entries. Since $s$ lies within a spatial distance of $\delta$ from $s'$, each row of transition matrices has only $n$ non-zero elements where $n \ll n$. In view of (1), $\tilde{n} = 5$. Therefore, to save time and memory, each $T_{n \times n}$ sparse matrix is reshaped into an $n \times 5$ matrix whose row elements refer to the index of successive states. For this to work, a coding procedure was employed to encode each four-tuple $(x_k, y_k, \theta_k, b_k)^T$ to an indexing integer ranging from 1 to $n$ and vice versa.

Considering the fact that an insertion results in some trauma to living tissue, $C_r$ is applied for every unit length of the inserted needle. Note that more damage occurs when the needle is rotated so $C_r$ is incurred after each rotation while $C_r > C_s$. Prohibitive costs of $C_o$ and $C_w$ are also added when collision with an obstacle or exiting the workspace. These parameters are initially set as $C_i = 10$, $C_r = 40$, $C_o = 200$, and $C_w = 100$. Finally, a summation of all individual costs gives the associated cost matrices or $C(s, \pm 1, s')$.

As the first test, a grid of $12 \times 8$ is employed to construct partitions. Fig. 2 demonstrates the order at which partitions were processed, and also dependencies of a sample partition. Inside each square, the number on the right hand side represents partition number while the number located on the left stands for the order of process. The latter number is set when a partition is processed for the iteration for the first time. At one glance, five partitions seem to be unprocessed that is a boon to the solver to ignore noninformative regions. Logically, there is no possibility for the needle to penetrate an obstacle; therefore, it is wise not to include the corresponding states in the algorithm. This idea is automatically realized by PPVI. Here, the red square depicts partition number 26 whereas its surrounding cyan area stands for $sdp(26)$.

Starting from the initial point selected as $t_0 = (0 2)^T$ and $\theta_0 = -30^\circ$, PPVI steers the needle toward the goal after 26 steps, which is shown in Fig. 3. Compared to the deterministic case as expected from [7], uncertain planning with the described parameters leads to a greater clearance from the obstacles with the larger number of turnings. For deterministic motion, $\beta_1$ is set to one and the rest of the noise coefficients are set to zero. By changing $C_r$ from 100 to 40, more emphasis is put on the maximum number of the needle turns. The results are shown in Fig. 4(a). By setting $C_i$ to 1 instead of 10, the results shown in Fig. 4(b) are obtained where the needle steers along a longer path and turns around the obstacles and maintains a good clearance from the closest obstacle to the entry point. In this test, the initial deflection was set to zero.

The main evaluation of PPVI is performed by measuring the amount of time required to find the optimal policy. Compared to the generic value iteration, PPVI exhibits a similar performance from the point of the resultant policy, but there is a considerable difference in terms of timing.

Using smaller partitions, it is possible to exclude a portion of centrally located obstacles from the planner’s effective workspace unlike the case in Fig. 2. Generally, PPVI saves more time by this selection as a result of dealing with a
smaller workspace. However, the trend demonstrating the behavior of PPVI performance with respect to the number or size of the partitions is too complex to be captured.

The numerical results are given in Table II. Roughly speaking, the computational burden induced by PPVI is $O(nk_1p(\frac{2}{p})^2) + O(k_2p)$. The first term illustrates the total complexity required for the individual partitions to be solved and the second term indicates the complexity of cross transitions. The notations $t_s$ and $t_c$ stand for the total time spent on solving the partitions and updating the transition priorities, respectively. As per section III, $m = 2$. Moreover, $k_1$ and $k_2$ are two other parameters whose values are determined by the number that a partition is referred to for being iterated and for the flow metric being updated, respectively.

In contrast with PPVI, the generic value iteration as implemented in [8] took around 8 hours and 4 minutes to converge after 196 full or 230,519,520 single updates. Accordingly, the update rate becomes roughly 7928.5 updates per second. A relatively high number of partitions, as shown by the last two rows of Table II, leads to an absolute drop in the effective update rate. This fact is justified as a result of the increased number of travels between the partitions that occupies more system overhead. According to our results, changing the flow metric will change the sequence of information propagation and also timing with no discernible effect on the optimal policy. In this problem, $H2$ slightly outperforms $H1$. Another factor affecting the timing performance is the partition size. In this respect, there is no solid theory to benefit from. Each partition contains $n_s = \frac{n}{n_x \times n_y}$ members. In the beginning, we do not know how to predict the optimal value of $n_s$, except to observe the outcome using a range of values for $n_x$ and $n_y$. We examined three sets as listed in Table II, and consequently the second selection for $(n_x, n_y)$ accompanied by $H2$ tends to yield the best results. Note that the absolute timings, listed in Table II, depend very much on the technique by which the algorithm is implemented. Developing a fast method to access the planner’s data base and designing an efficient way to check a collision are among the programming skills that enable us to accelerate PPVI more.

VI. ACKNOWLEDGMENTS

This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grants RGPIN-1345 and RGPIN-346166.

REFERENCES


