Xiaoshan Pan, Civil and Environmental Engineering, Stanford University

Wheelchair Robotic Motion Planning

1. Introduction

Accessibility is one key aspect of building design, which is legally enforced for most public facilities in the United States and many other countries. Extensive guidelines have been developed to ensure the accessibility of buildings, such as the Americans with Disabilities Act Accessibility Guidelines (ADAAG), the Fair Housing Accessibility Guidelines (FHAG), and the Uniform Federal Accessibility Standards (UFAS).

The traditional method of assessing the accessibility of building design is the prescriptive code-based approach, which applies the interpretation of codes to designs in order to ascertain whether or not there has been code compliance. However, Kiliccote (1996) argues that prescriptive-based codes can be ambiguous, contradictory, complex, and restrictive. Furthermore, Han et al. (2001) claim:

> Solutions constrained by prescriptive-based codes such as the ADAAG address only a fraction of the possible solutions that meet the design intent or objectives of these codes. Since it often is implicit, it is often difficult for both designers and code checkers to discern the design intent and objectives of a building code or code provision… instances exist in which adhering to these prescriptive provisions produces a design that may not be usable. (p. 1)

As an alternative, the performance-based approach was presented by Han et al. (2002), which evaluates the accessibility of a facility using motion-planning techniques. This approach directly simulates the behaviors of a moving wheelchair under the constraints of the wheelchair itself, geometrical spaces, and building codes for accessibility analysis; thus, the performance-based approach is able to provide direct, intuitive, and unambiguous results.

While the performance-based approach has been shown to be successful by Han (2000), Han et al. (2001), and Han et al. (2002), the motion planner that it utilizes has limited potential. For example, the work of Han et al. (2002) constrains the wheelchair motion to three options - left, right, and forward - despite the fact that all wheelchairs are able to roll backward; this constraint precludes the performance-based approach from assessing the cases in which backward motions are essential, such as the T-Shape Space (See Figure 1) in which a wheelchair must back up in order to make a 180 degree turn. In addition, the motion planner demonstrates low performance in the case of running against

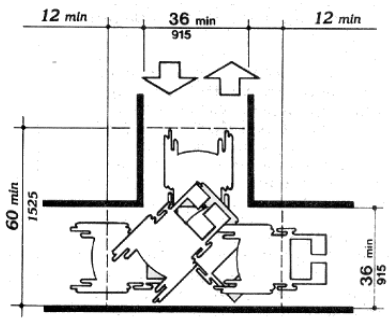


Figure 1: An example showing that backward motions are essential for simulating the typical motion of a wheelchair. (http://www.access-board.gov/adaag/html/figures/fig3b.html)

large-sized inputs, thus precluding the specific implementation from being applied to broader applications.

Many path-finding techniques have been developed in the field of motion-planning research, such as potential field, road map, and cell decomposition. However, none of these solves the planning problem in its full generality (Latombe, 1991). Depending on the nature of the problem, some techniques work better than others. Particularly, a desired motion planning technique that we are seeking should be able to: 1) handle nonholonomic constraints, (i.e., the kinematic constraints of a wheelchair are nonholonomic); 2) plan a path in real time (i.e., the faster the planning technique is, the more practical the performance-based accessibility checking approach would be). One motion planning tool, the Rapidly-exploring Random Tree (RRT) (LaValle, 1998), is suitable for solving nonholonomic planning problems, and it employs randomization to explore large state spaces efficiently; to date, RRT has been employed by many researchers for solving many practical path planning problems (Branicky and Curtiss 2002; Bruce and Manuela, 2002; Kuffner et al., 2002; Liu and Badler, 2003); therefore, the authors have chosen it to be the basis for our wheelchair robotic simulation.

LaValle (1998) introduced the Rapidly-exploring Random Tree (RRT) as a planning approach to quickly search high-dimensional spaces with both algebraic constraints (arising from obstacles) and differential constraints, which can be applied to a wide variety of planning problems with nonholonomic constraints. However, because RRT is a general-purpose technique, substantial efforts are required to develop a suitable planner to foster the performance-based accessibility.

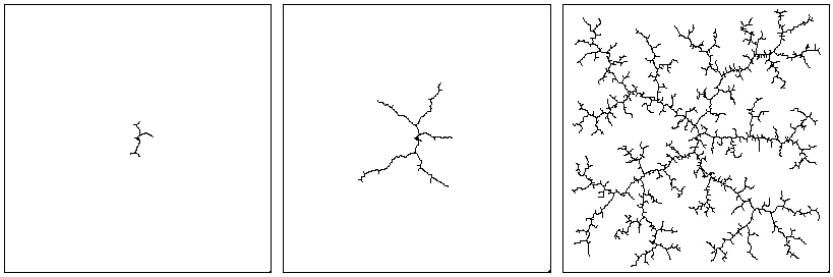


Figure 2: An RRT rapidly explores the uncovered region before uniformly spreading over the entire space (LaValle, 1998, p. 3).

This paper presents a computational wheelchair modeling process, particularly emphasizing the development of an RRT-based planner to undertake wheelchair motion planning. First, we give a description of how the geometric and kinematic constraints of a wheelchair are modeled. Second, we present the development of an RRT-based planner to simulate wheelchair motions. We then introduce a computer tool, a wheelchair-accessible design assistant (WADA), in which the wheelchair simulation model is utilized; in addition, two example applications of WADA are provided – evaluating accessibility guidelines and assisting barrier-free designs. Finally, we present the conclusions.

In order to construct a computational model of a wheelchair, three factors need to be considered: 1) the geometry of the wheelchair, 2) the kinematic constraints of the wheelchair - possible ways that the wheelchair can maneuver, and 3) a motion planning algorithm that is able to compute an obstacle-free path if an initial and a goal position of the wheelchair are given. The factor 1) and 2) formulate the representation of a wheelchair, while 3) facilitates the representation.

2. Modeling the Geometry and the Motion of a Wheelchair

The first issue of modeling a wheelchair is to represent its geometry mathematically. Even though a wheelchair is 3-dimentional in reality, for the sake of reducing the complexity, at this stage we only take the dimensions of its plan view into consideration. Thus, the geometry of a wheelchair can be simplified as a polygon – a 2D robot (see Fig. 3), and different types of wheelchairs can be represented by various sized polygons flowing this approach.

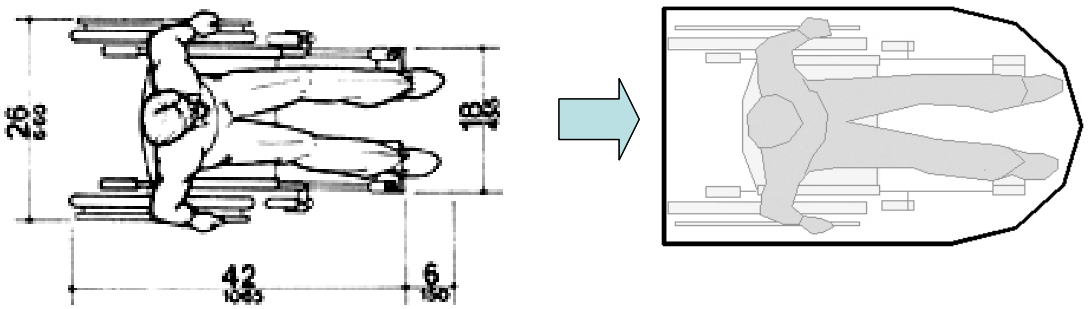


Figure 3: The geometry of a wheelchair is simplified as a polygon.

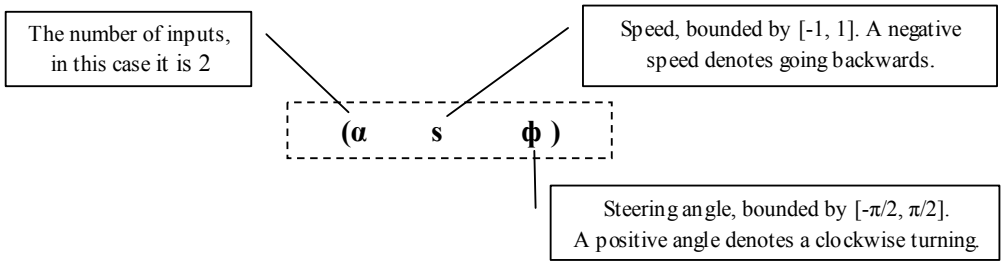


Figure 4: The format of an input vector.

The second, we need to formally represent the motions of a wheelchair. A wheelchair's motion must comply with two types of constraints: geometric constraints and kinematic constraints. The geometric constraints forbid a wheelchair from colliding with other obstacles in the space in which it maneuvers, which is typically accomplished by conducting collision detection to configurations of the wheelchair. The kinematic constraints define how new wheelchair configurations are derived from old ones; some of the associated parameters are velocity, steering angle, and turning radius, and they can be modeled mathematically by applying a set of input vectors to some state transition equations. We give a detailed discussion of input vectors and state transition equations in the next section.

2.1. State Transition Equations and Input Vectors

A wheelchair moves under kinematic constraints (e.g., it can move straight forward or straight backward, turn left or right, but it cannot move sideways); these constraints can

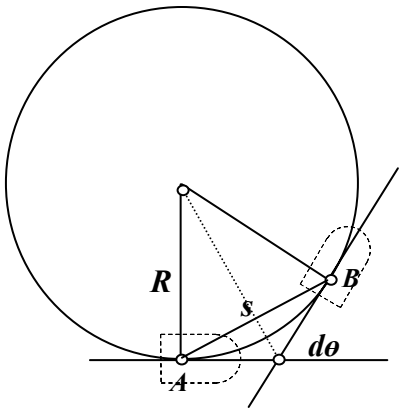


Figure 5: The geometrical relationship between the turning radius and the steering angle of a wheelchair;
(*A* – current state; *B* – new state; *R* – turning radius; *s* – speed;
dθ - the velocity of facing direction, and $d\theta = s/L * \tan\phi$).

be represented mathematically using the notions of state transition equations and input vectors. Let each state of a wheelchair be represented as **(x, y, θ)**, where **(x, y)** denotes the coordinates of the wheelchair in a 2D space; **θ** denotes the direction that the robot is facing. Then, we can define a set of state transition equations that indicates how the state of the wheelchair changes over time, given a current state and current input (e.g., the steering angle of the wheelchair). LaValle (2001) presented such a set of state transition equations as follows:

$$dx = s*\cos\theta$$
$$dy = s*\sin\theta$$
$$d\theta = s/L*\tan\phi$$

The speed of a wheelchair is represented as **s**; the distance between the front and rear wheels is represented as ***L***, and the steering angle is denoted by **ϕ.** An input vector has the format shown in Figure 7. For example, a set of input vectors

2 1.0 0.0
2 1.0 0.4
2 1.0 -0.4

denote that each state of a wheelchair can derive three new states by applying these inputs to the above state transition equations for:

1. going straight forward,
2. making a right turn while going forward, and
3. making a left turn while going forward.

```
RRT-ExtExt(x_init, x_goal)
    Tree_a.init(x_init); Tree_b.init(x_goal);          // start two RRT trees
    For k = 1 to k do                                  // a loop that iterate K times
        x_rand ← RANDOM_STATE();                       // pick a random sample in space
        If not (EXTEND(Tree_a , x_rand) = Trapped) then // see if the tree can reach the sample
            If (EXTEND(Tree_b, x_new) = Reached) then  // see if two trees can be connected
                Return PATH(Tree_a ,Tree_B);           // return a path if two trees are connected
            SWAP(Tree_a ,Tree_B);                      // swap two trees
    Return failure;                                    // quit after K iterations

EXTEND(Tree , x)
    x_near ← NEAREST_NEIGHBOR(x, Tree);                // find a node that is the nearest to x
    If NEW_STATE(x, x_near, x_new, u_new) then         // create x_new from x_near, try to reach x_new
        Tree.add_vertex(x_new);                        // add x_new to the tree
        Tree.add_edge(x_near, x_new, u_new);           // add a new edge between x_near and x_new
        If x_new= x, then                              // test to see if x_new overlaps with x
            Return Reached;                            // x has been reached (x can be the goal)
        Else
            Return Advanced;                           // tree has been extended
    Return Trapped;                                    // tree extension fails
```

Figure 6: The RRT-ExtExt (LaValle and Kuffner, 2000)

If the turning radius of a wheelchair is known, instead of the steering angle, we can compute the corresponding steering angle using the following formula:

$$\phi = \arctan((2L/s)* \arcsin(s/(2R)))$$

R denotes the turning radius of a wheelchair. The derivation of the formula is straightforward, based on the geometric relationship depicted in Figure 5.

3. The Development of a Wheelchair Motion Planner

The motion planner that we have developed is based on an original RRT planner – RRT-ExtExt – presented by LaValle and Kuffner (2000). In this section, we first give an overview of the original RRT-ExtExt planner; we then discuss how we have enhanced it to adopt wheelchair motions.

3.1. The RRT-ExtExt

A planning problem can be formulated in terms of six components:

1. State Space: a topological space, X
2. Boundary Values: $x_{init} \in X$ and $x_{goal} \in X$
3. Collision Detector: a function, $D : X \rightarrow$ {true, false}, that determines whether global constraints are satisfied from state x.

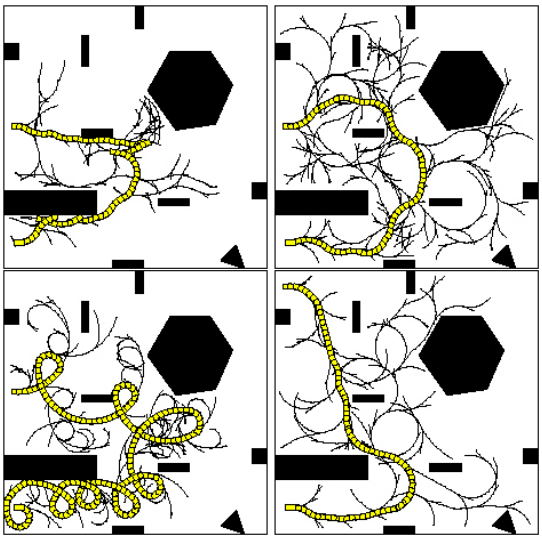


Figure 5: Several car-like robotic nonholonomic problems solved by RRT-ExtExt (LaValle and Kuffner, 2000, p. 12).

4. Inputs: a set, U, which specifies the complete set of controls or actions that can affect the state.
5. Incremental Simulator: given the current state, x(t), and inputs applied over a time interval $\{ u(t`) \mid t \leq t` \leq t + \Delta t \}$, compute $x(t + \Delta t)$.
6. Metric: a real-valued function, $\rho : X * X \rightarrow [0, \text{infinity})$, which specifies the distance between pairs of points in X. (LaValle and Kuffner, 2000, p. 2)

The RRT-ExtExt starts by growing two RRT trees – one from x_{init} and the other from x_{goal}; a solution is found if the two trees meet. Due to its randomized approach, both RRT trees rapidly extend to the unexplored state space and pull themselves towards each other. (See Figure 6 for the detailed algorithm.)

The dual tree approach of RRT-ExtExt generally performs better than a single tree approach; its successes in solving nonholonomic planning problems have been demonstrated by LaValle and Kuffner (2000) (see Figure 5).

However, since RR-TExeExt is indeed a randomized incremental planner, it inherits the shortcomings that most randomized approaches have. For example:

1. It is difficult to evaluate the performance of the planner if it runs against large-sized inputs; the time it takes can be short, long, or anything in between. The probabilistic distribution depends on the nature of the problem.
2. If there are several narrow passages in the configuration space between an initial state and a goal state, then the probability of the planner finding a path that crosses

narrow passages is rather low. (This problem is partially caused by the random sampling technique that it utilizes, in addition to the geometric and kinematic constraints of the problems).

In the next section, we give a brief discussion of how a pure randomized sampling technique is insufficient when solving problems in the context of wheelchair motion planning that contains narrow passages,

3.2. The problem of Randomized Sampling under Narrow Passage Conditions

As we have shown in Figure 2 (page 3), an RRT-based planner can rapidly explore state spaces; however, this no longer holds true if it must find a path that crosses several local spaces connected via narrow passages created by the kinematic constraints of a wheelchair. The reason for this deceleration is that the random sampling technique a RRT-based planner employs does not bias to obtain more samples near regions around narrow passages in order to extend through the passages quickly. (See Figure 6 on page 9 for a conceptual depiction).

Even though, by nature, a narrow passage problem is difficult to solve, it is particularly useful to develop promising solutions suitable for tackling motion planning for wheelchairs since it is common for a wheelchair to encounter narrow passages in real applications. For instance, Figure 1 (page 2) exhibits a T-Shape space in which a wheelchair needs to make a 180 degree turn. Under the constraints of the given space dimensions, there are only two possible routes that a wheelchair can use to accomplish the objective:

1. Going forward → turning left → going backward → going forward → turning left;

or

2. Going forward → turning right → going backward → going forward → turning right.

To execute either of the two routes, a wheelchair must follow the same sequence of actions – going forward, going backward, then going forward; there is no other alternative. Thus, the search space for this example can be conceptualized as three local C-spaces connected via two narrow passages (see Figure 7). It is worthwhile noting that

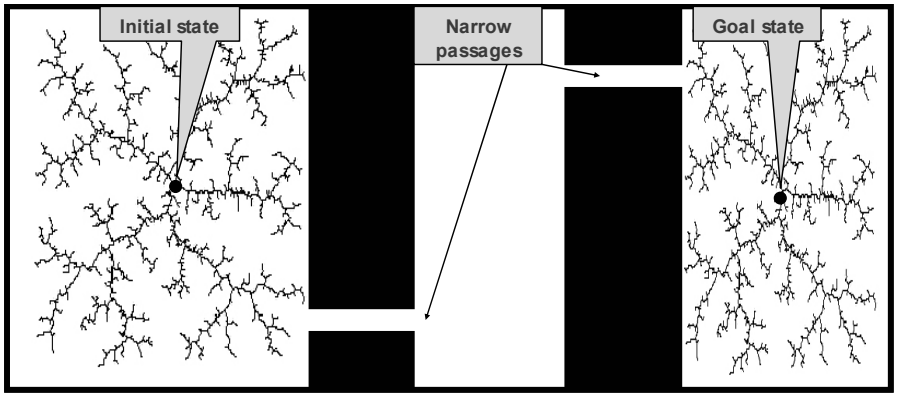


Figure 6: The extension of RRT trees are blocked by narrow passages.

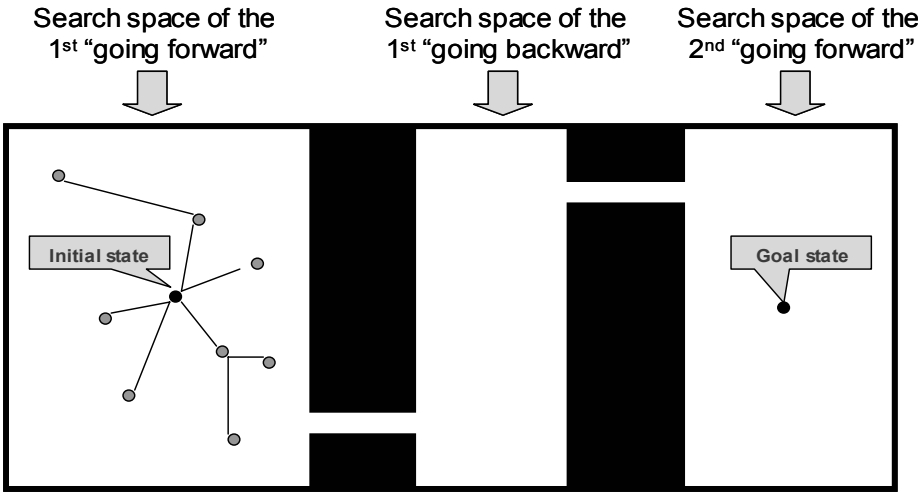


Figure 7: The search space of a wheelchair in a T-shape space.

the kinematic constraints of the wheelchair contribute the most to these narrow passages in addition to the geometric constraints of the spatial dimensions.

Based on the authors' experiment, the problem described above has been shown to be surprisingly difficult; none of the original RRT-based planners was able to solve it even if the given timeline is as long as several hours. The reason is simple: before an RRT tree can grow near the narrow passages, all the sampling efforts from other local spaces are wasted.

As has been shown in Figure 7, the narrow passages are located at the places where a wheelchair performs a motion switch – changing its direction from forward to backward, or vice versa. Therefore, one observation is: if a planner has control over its motion switch (instead of depending on randomness), it allows the RRT trees to approach the

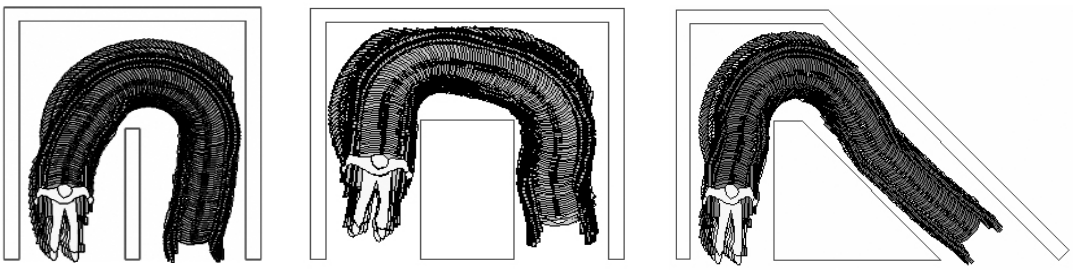


Figure 8: Some example wheelchair path planning problems solved by using forward-only motion.

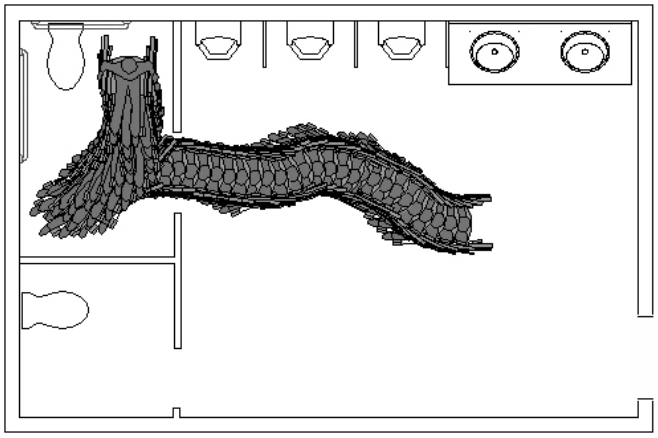


Figure 9: The bathroom problem that requires a wheelchair to back up at least once.

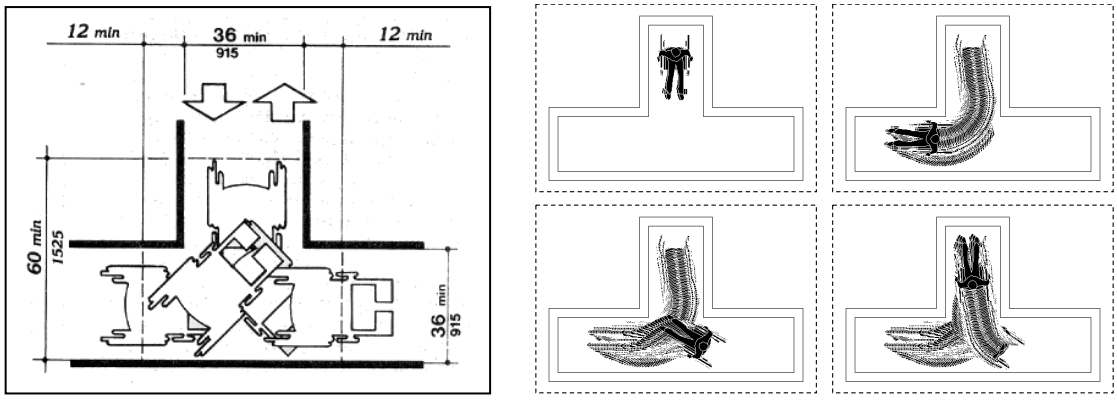
entries of a narrow passage before it takes samples from other local spaces, then enormous computation efforts caused by random sampling will be eliminated.

Based on this observation, we have developed an enhanced RRT-ExtExt planner, which is able to solve narrow passage problems more efficiently.

3.3. The Enhanced RRT-ExtExt

The basic ideas of the enhanced RRT-ExtExt include the following:

1. Instead of using one wheelchair robot that maneuvers both forward and backward to expand an RRT tree, the new planner utilizes two wheelchair robots, which have the same geometric constraints, but different kinematic constraints (i.e., one goes forward only, and the other goes backward only);



Wheelchair Turning Space
T-Shaped Space for 180 Degree Turns
(ADAAG)

Trajectory Generated by Motion Planner

Figure 10: Wheelchair motion planning for a T-shape space.

2. The planner employs a controlling mechanism that switches between these two robots (i.e., one at a time) for the purpose of eliminating unnecessary computation while searching through the C-spaces.

Based on these ideas, we present an enhanced RRT-ExtExt algorithm for solving narrow passage problems in the following section.

3.3.1. The Algorithm

The enhanced RRT-ExtExt starts two wheelchair robots – one begins at the initial position and is constrained to maneuver forward only (i.e., move straight forward, turn left, or turn right); the other begins at the goal position and is constrained to maneuver backward only. This setting guarantees that the planner only searches a local C-space created by the forward-only constraint; many wheelchair motion planning problems can be solved using this setting (see Figure 8).

However, if a path has not been found even after the RRT trees have achieved good coverage over the local C-space (which may imply the existence of narrow passages), the planner switches the robots to backward-only ones, while maintaining the existing RRT trees, then continues to search in the local C-space created by the backward-only constraints. Remarkably, this strategy increases the probability of quickly expanding the RRT trees from one local C-space into another because: 1) the good coverage of the trees in the previous local C-spaces renders better opportunities for a tree to locate and go

```
ENHANCED-RRT-ExtExt(x_init, x_goal, K, N)
    Tree_a.init(x_init); Tree_b.init(x_goal);          // start two RRT trees
    Motion_mode = Forward;                             // set initial motion as forward-only
    For n = 1 to N do                                  // control the number of motion switch
        For k = 1 to K do                              // a loop that iterate K times
            x_rand ← RANDOM_STATE();                   // pick a random sample in space
            If not (EXTEND(Tree_a , x_rand) = Trapped) then   // see if the tree can reach the sample
                If (EXTEND(Tree_b, x_new) = Reached) then     // see if two trees can be connected
                    Return PATH(Tree_a ,Tree_B);              // return a path if two trees are connected
                SWAP(Tree_a ,Tree_B);                         // swap two trees
        SWITCH_MOTION(M_mode);                         // switch motion
    Return failure;                                    // quit after K switches
```

Figure 11: The enhanced RRT-ExtExt

through local passages rapidly; 2) before a RRT tree can grow near the narrow passages, no efforts are wasted in terms of sampling from other local spaces (as the original planner would do).

Under the control of a user, the planner may repeat the motion switch as many cycles as necessary until a path is found, or abort the search if a problem is found to be intractable. Another category of wheelchair path planning problems can be solved quickly by using this approach, such as the Bathroom problem that requires one motion switch and the T-shape space problem that requires two motion switches (see Figure 9 and Figure 10 on page 10 and 11).

The algorithm is given in Figure 11. **K** represents the maximum number of extensions that a RRT tree executes in order to achieve good coverage of a local C-space; **N** denotes the number of motion switches that a user would like the planner to undertake. By using motion switches, the planner extends a RRT tree from one local C-space to another through narrow passages until a solution is found. The random sampling technique is utilized for each local C-space, but the overall expansion of a RRT tree crossing different local C-spaces is systematic; this hybrid approach has been shown to be effective for solving wheelchair motion planning problems.

4. Example Applications

Based on the Enhanced RRT-ExtExt introduced above, we have developed a computer tool – the wheelchair accessible design assistant (WADA) – which integrates the

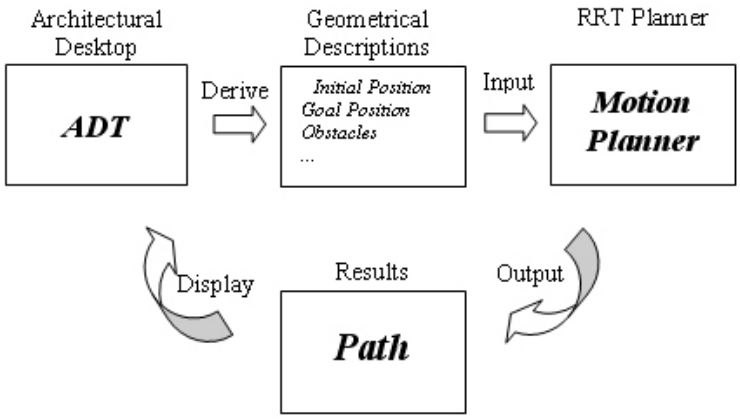


Figure 12: Information flow of the WADA.

Architectural Desktop (ADT) and Enhanced RRT-ExtExt for assessing wheelchair accessibility of an architectural space. WADA works as follows: at first, it derives the geometric information of an architectural space directly from ADT and then sends it to the motion planner (together with wheelchair descriptions and a set of input vectors that represent nonholonomic constraints of the wheelchair); the planner determines whether or not the space is accessible and sends results back to ADT (see Figure 12).

Two example application areas of WADA are 1) evaluating accessibility guidelines and 2) assisting barrier-free design.

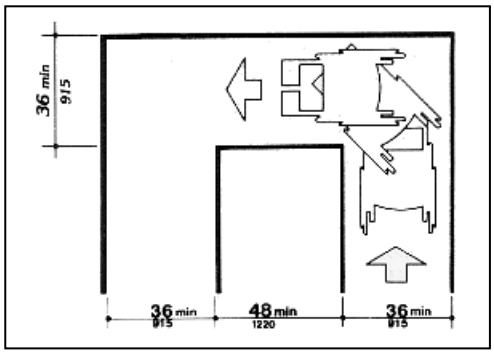
4.1. Evaluating Accessibility Guidelines

The WADA is significantly useful in terms of facilitating a performance-based approach for evaluating accessibility guidelines, because the WADA makes it convenient to formulate testing cases based on code descriptions, and then evaluate them using the planner.

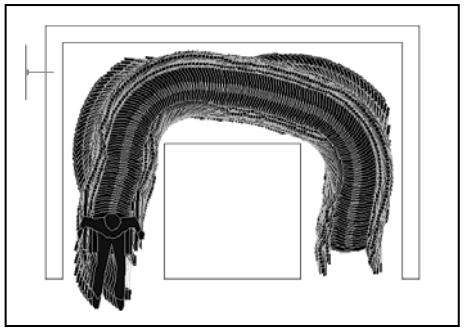
As the first example, in order for a wheelchair to make a 180 degree turn in a T-shaped space, the corresponding ADAAG code states:

> The T-shape space is 36 inches (915 mm) wide at the top and stem within a 60 inch by 60 inch (1525 mm by 1525 mm) square. (http://www.access-board.gov/adaag/html/figures/fig3b.html)

After a corresponding case has been built using ADT, the result (i.e., a wheelchair path) produced by WADA establishes the validity of the requirements described by the code (see Figure 10 on page 11).



Accessible Route
90 Degree Turn (ADAAG)



Trajectory Generated by Motion Planner

Figure 13: Motion planning for a 90 degree turn.

As another example, for a 90 degree turn space, the corresponding ADAAG code says:

> A 90 degree turn can be made from a 36 inch (915 mm) wide passage into another 36 inch (915 mm) passage if the depth of each leg is a minimum of 48 inches (1220 mm) on the inside dimensions of the turn. (http://www.access-board.gov/adaag/html/figures/fig7a.html)

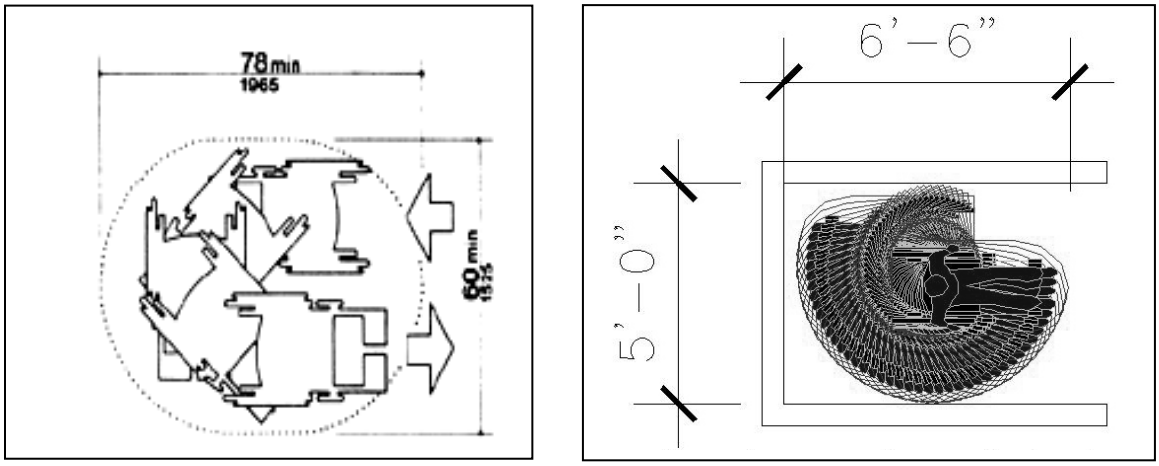
Figure 13 demonstrates that the result produced by the WADA agrees with the code descriptions.

The WADA is also helpful in identifying the shortcomings that exist in the codes. For example, in order for a wheelchair to make a smooth U-turn, ADAAG states the requirement of a space to be the following:

> The space needed for a smooth U-turn in a wheelchair is 78 inches (1965 mm) minimum by 60 inches (1525 mm) minimum. (http://www.access-board.gov/adaag/html/figures/fig3a.html)

According to the results shown in Figure 14 (page 15), a smooth U-turn is possible only if a wheelchair has a turning radius of less than six inches; in order to make the turn, one wheel of the chair must roll backward while the other rolls forward. Therefore, the motion depicted by ADAAG in Figure 14 (i.e., the picture on the left) is rather misleading.

As we have seen from the above examples, the WADA can significantly improve the efficiency and the quality of the performance-based code evaluation.



Space Needed for Smooth U-turn in a Wheelchair (ADAAG)

Wheelchair Trajectory

Figure 14: Motion planning for a smooth U-turn.

4.2. Assisting Barrier-Free Design

Another practical use of the WADA is to assist architects in designing barrier-free architectures. One possible scenario would be:

1. An architect develops a floor plan using the ADT and wants to check if the design is indeed satisfied by the accessibility requirements for wheelchairs;
2. The architect applies the floor plan to the WADA, along with a pair of designated locations which are expected to be accessible, and the WADA reports the results back within a short time;
3. Based on the results, the architect modifies the design until the corresponding requirements are satisfied, (when necessary, repeat 2 and 3).

Figure 15 (page 16) illustrates a wheelchair access path generated by the WADA, which demonstrates the accessibility of the design (from the entry ramp to the bedroom).

5. Conclusions and Future Research

We have presented the development of an enhanced RRT-based motion planner, which adopts a hybrid approach – random sampling in local C-space, plus systematically expanding an RRT tree crossing different C-spaces separated by narrow passages – to undertake wheelchair motion planning. The resultant planner has demonstrated

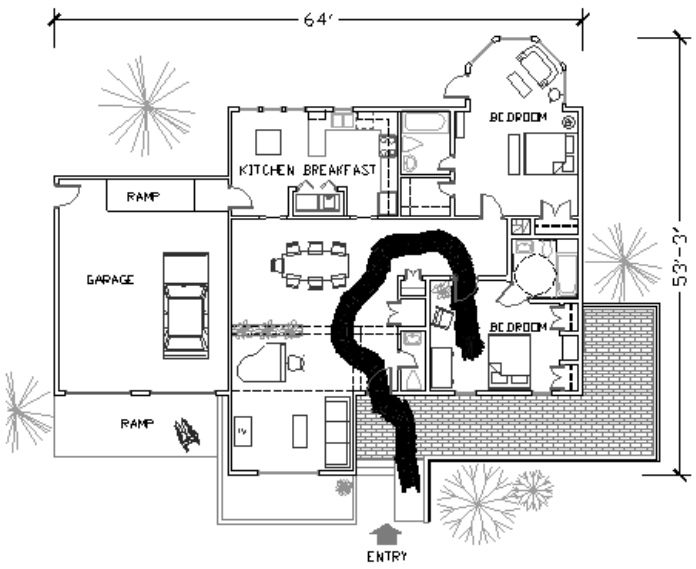


Figure 15: A wheelchair access path generated by the WADA.

significant capability of solving wheelchair motion planning problems. Based on this enhanced RRT planner, we have constructed a computer tool – WADA – to facilitate the performance-based building accessibility assessment and barrier-free design.

One future study for this project is to develop a method of detecting the coverage of a RRT tree in a local C-space in real time. One critical condition for the current planner to locate the position of a narrow passage is to expand an RRT tree far enough so that the tree provides a good coverage of the local C-space. Insufficient coverage results in difficulties in searching for narrow passages, while over-growing a tree would result in unnecessary computation time. Currently, determining when to cease expanding a RRT tree is accomplished through assigning a pre-determined value to K (i.e., the maximum number of extensions of a RRT tree, see Figure 11), assuming that a RRT tree will supply enough coverage in a local C-space when K is reached. However, such an assumption is insufficient to guarantee the efficiency of the algorithm. Researching a method that measures the coverage of a tree would significantly enhance the robustness and performance of the planner.

References:

Branicky, M. and Curtiss, M. (2002). *Nonlinear and Hybrid Control Via RRTs*. Proc. Intl. Symp. on Mathematical Theory of Networks and Systems. South Bend, IN, August.

Bruce, J. and Manuela, V. (2002). *Real-Time Randomized Path Planning for Robot Navigation. Proc. IROS-2002*, Switzerland, October.

Han, C., Law, K., Latombe, J-C., and Kunz, J. (2002). *A Performance-Based Approach to Wheelchair Accessible Route Analysis.* Advanced Engineering Informatics, 16:53-71.

Han, C., Kunz, J., and Law, K. (2001). *Compliance Analysis for Disabled Access*, in Advances in Digital Government Technology, Human Factors, and Policy. William J. McIver, Jr. and Ahmed K. Elmagarmid (eds). Boston: Kluwer.

Kiliccote, H. (1996). *A Standards Processing Framework.* PhD Thesis, Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, PA.

Kuffner, J., Kagami, S., Nishiwaki, K., Inaba, M., and Inoue, H. (2002). *Dynamically-stable motion planning for humanoid robots.* Autonomous Robots (special issue on Humanoid Robotics)*, 12:105-118.*

Latombe, J. (1991). *Robot Motion Planning.* Boston: Kluwer Academic Publishers.

LaValle, S. (2001). *Motion Strategy Library, Version 1.2*. Computer Science Dept., University of Illinois. Available: http://msl.cs.uiuc.edu/msl/

LaValle, S., Kuffner, J. (2000). *Rapidly-Exploring Random Trees: Progress and Prospects by. Proc*. 2000 Workshop on the Algorithmic Foundations of Robotics.

LaValle, S. (1998). *Rapidly-exploring Random Trees: A New Tool for Path Planning.* TR 98-11, Computer Science Dept., Iowa State University.

Liu, Y. and Badler, N. (2003). *Real-Time Reach Planning for Animated Characters Using Hardware Acceleration.* Computer Animation and Social Agents, IEEE Computer Society, New Brunswick, NJ, May 2003, pp. 86-93.