USING MOTION-PLANNING TO DETERMINE THE EXISTENCE OF AN ACCESSIBLE ROUTE IN A CAD ENVIRONMENT

Xiaoshan Pan, M.S., Stanford University

Charles S. Han, Ph.D., Stanford University

Kincho H. Law,[1] Ph.D., Stanford University

Department of Civil and Environmental Engineering
Terman Engineering Center, Room 242
Stanford University, Stanford, CA 94305-4020

[1] Tel: 650-725-3154, Fax: 650-723-7514, law@stanford.edu

FOOTNOTES

This research is supported by the National Science Foundation under Grant No. EIA-9983368 and by the Center for Integrated Facility Engineering at Stanford University. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsoring agencies. The authors would like to thank Mr. Joe Cavanaugh for his participation in this research.

ABSTRACT

We describe an algorithm based on motion-planning techniques to determine the existence of an accessible route through a facility for a wheeled mobility device. The algorithm is based on LaValle's work on Rapidly-exploring Random Trees (RRT) and is enhanced to take into consideration the particularities of the accessible route domain. Specifically, the algorithm is designed to allow performance-based analysis and evaluation of a facility. Furthermore, the parameters of a wheeled mobility device can be varied without recompilation thus allowing standards writers, facility designers, and wheeled mobility device manufacturers to vary the parameters accordingly. The algorithm has been implemented in a computer tool that works within a computer-aided design and drafting (CADD) environment.

KEYWORDS

Accessibility, wheeled mobility device, motion planning, simulation, performance-based methods, computer-aided design and drafting

1. INTRODUCTION

The intent of accessibility guidelines such as the Americans with Disabilities Act (ADA) Accessibility Guidelines (ADAAG) is to provide the same or equivalent access to a building and its facilities for disabled persons (for example, persons restricted to a wheeled mobility device, persons with hearing and sight disabilities) and persons without qualifying disabilities. To fulfill this intent, the authors of the Americans with Disabilities Act (ADA) have developed prescriptive measures such as various clearances and reach thresholds for building components. For example, the ADA has developed guidelines for minimum clearances to allow transfer of a person from a wheeled mobility device to a toilet and minimum lengths of grab bars associated with a toilet. Prescriptive statements are formulated to establish concrete tests for many of such measures.

Modeling these prescriptive provisions and using them to analyze a facility is often a trivial task. However, there are two limitations to a prescriptive analysis framework to analyze a facility against guidelines such as the ADAAG:

1. Using prescriptive provisions can lead to problems such as conflicting and ambiguous statements, thus making the code provisions difficult to parse not only by computers, but for humans as well. A design that fulfills a set of prescriptive provisions does not always imply usability. Conversely, a design that does not meet a set of prescriptive previsions could actually be accessible by a person in a wheeled mobility device.

2. Even if a set of prescriptive provisions is not conflicting or ambiguous, formulating computational methods for certain provisions can be overly complex.

To address these limitations, a simulation-based approach can be used as long as the modeled simulation captures the intent and the parameters of a given set of provisions. The nature of the accessible route problem makes it an ideal candidate for the development of a simulation tool using motion-planning

techniques. In an earlier work, we proposed a hybrid prescriptive- and performance-based framework that combined prescriptive methods that addressed the component-based accessible route provisions and motion-planning based simulation techniques to determine the paths composing an accessible route (Han 2002). Furthermore, we have demonstrated an online code-checking framework that transfers building model data from a CAD environment, Autodesk's AutoCAD to an accessibility analysis Web Service using an intermediate building model, the International Alliance for Interoperability (IAI) Industry Foundation Classes (IFC) (IAI 1997), to transfer the design data over the Internet (Han 1998). In this paper, we present an enhanced motion-planning algorithm to solve the general accessible route problem. The motion-planning-based accessible route analysis is implemented directly in a CAD environment, Autodesk Architectural Desktop (ADT) using no intermediate building model and independent of Internet connectivity. Last but not least, we present a number of examples to illustrate the potential applications of the performance-based simulation techniques for accessibility analysis.

2. MOTION-PLANNING TECHNIQUES FOR THE ACCESSIBLE ROUTE

2.1 Background

Focusing on wheeled mobility device access, persons in wheeled mobility devices must be provided the same or equivalent access to a building and its facilities as persons who do not use such devices. While prescriptive accessibility checking may be possible for individual building components, global issues of accessibility require directly capturing the design intent of a set of provisions. "Equivalent" access is somewhat ambiguous, but the intent is that a person in a wheeled mobility device need not go through extreme methods to be able to have access to a building's facilities. For example, if a person not using a wheeled mobility device needs to travel a certain distance to a bathroom facility, then a person using a wheeled mobility device should have to travel approximately the same distance to use either the same or an equivalent bathroom facility. The concept of access is a system-wide issue related to the entire floor or building as well as a local issue confined within a defined space. Methods can be developed to analyze

local prescriptive issues such as clearances around building components. However, testing for compliance of global issues such as the existence of an accessible path can be more easily done using performance-based simulation techniques.

Simulation can be employed to address provisions that are difficult to analyze statically such as the existence of an accessible path in a building design. Accessible route provisions examine global issues of a project as opposed to looking at localized phenomena. In examining the issues of accessibility, the design analysis program must consider accessible path. For example, if a door is on an accessible path, the program can check its necessary clearances. However, since these are local and static checks, the program cannot guarantee that in moving from one room to another, even if individual doors meet the code, a disabled person can actually have access to these doors.

Simulation of a wheeled mobility device moving through the space is a logical approach. Using motion planning, a research area in robotics, such a simulation is possible (Latombe 1991). The performance-based approach was presented by Han (2002) which introduces a method to determine 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, the geometry of the spaces, and guidelines for accessibility analysis. Thus, the performance-based approach is able to provide direct, intuitive, and unambiguous results. The path-planning techniques explored in this previous research also have the advantage over prescriptive-based methods by being parameterized. Both the dimensions of the wheelchair and the turning constraint parameters are adjustable inputs to the motion planner thus reflecting preferences of the individual wheelchair user.

The approach in the earlier study was to combine prescriptive methods to check the accessibility of specific components (such as doors and openings) and the motion-planning techniques to verify the existence of an accessible path between these components. Figures 1 to 4 show an analysis of a university facility using the hybrid framework. This specific example analyzes the accessibility of a toilet

facility from the entrance of the building. Figure 1shows the current configuration of the first floor of the facility. Figure 2 shows the results of the analysis of the floor according to the prescriptive provisions of the ADAAG (1997). Note that the results are presented in a frame-based web page that links a graphical representation of the floor, and violation comments, and the ADAAG. The graphical violation(s) are hyperlinked to the comments which are in turn hyperlinked to the relevant ADAAG provisions. In this case, the partition walls around the water closet are the cause of the violation. In Figure 3, a user validates the lack of accessibility of the toilet facility. Once the violation is corrected (by removing the partition walls), the framework informs the user that there is indeed an accessible route. In addition to the prescriptive based analysis, Figure 4 shows the results of the motion-planning-based analysis. Note that the route is discontinuous at the doorways as the dimensions of these components are checked in accordance to code-specified measurements separately and the final result is a concatenation of the route segments through valid components.

While the performance-based approach has been shown to be successful by Han (2002), the motion planner that was utilized has limited potential. For example, wheelchair motion is constrained to three options (left, right, and forward) despite the fact that many wheeled mobility devices 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 5) in which a wheeled mobility device must back up in order to make a 180-degree turn. In addition, the motion planner is computationally expensive in the case of running against 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 the techniques can solve 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:

- handle nonholonomic constraints, i.e., the kinematic constraints of a wheeled mobility device are nonholonomic

- plan a path in real time e.g., 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 2002, Bruce 2002, Kuffner, 2002, Liu 2003). 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 analysis.

This paper presents the development of an RRT-based planner to simulate wheeled mobility device for an accessibility assessment. First, we give an overview of one of the original RRT-based planners—RRT-ExtExt. Second, we describe how we have enhanced the motion planner and adapted the planner for wheeled mobility device motions and constraints.

2.2 The Original RRT Planners

The key idea of the RRT is to bias the exploration toward unexplored portions of the space by taking random samples in the state space and incrementally pulling the search tree toward them. This procedure causes the RRT to rapidly explore the uncovered region before uniformly spreading over the entire space (see Figure 6).

Lavalle (2000) introduced two categories of RRT-based planners: single-RRT planners and bidirectional planners. Single-RRT planners start the search trees at initial positions and then grow the search trees towards goal positions; two of the instances are RRT-GoalBias and RRT-GoalZoom. Bidirectional-RRT planners have been inspired by classical bidirectional search techniques (e.g., bidirectional A* algorithms), which grow two search trees – one from an initial position and the other from a goal position; a solution is determined once the two trees meet. Two examples of bidirectional-RRT planners are the RRT-ExtCon, and the RRT-ExtExt. Among these variants, LaValle (2002) suggested that the RRT-ExtExt is the best for nonholonomic problems.

A planning problem can be formulated in terms of six components (LaValle 2000):

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.
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, $\rho : X * X \rightarrow [0, infinity)$, which specifies the distance between pairs of points in X.

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. Figure 7 provides the details of the 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 as shown in Figure 8 (La Valle 2000).

However, since RRT-ExeExt is indeed a randomized incremental planner, it inherits the shortcomings that most randomized approaches would have. For examples:

1. It is difficult to evaluate the performance of the planner in the case of running 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 wheeled mobility device motion planning that contains narrow passages.

2.3 Randomized Sampling under Narrow Passage Conditions

As we have shown in Figure 8, an RRT-based planner can rapidly explore state spaces; however, this does not hold true if the planner must find a path that crosses several local spaces connected via narrow passages created by the kinematic constraints of a wheeled mobility device. The reason for this narrow passage problem 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 9 for a conceptual depiction).

Even though a narrow passage problem is difficult to solve, the problem is particularly important to develop solutions suitable to tackle motion planning for wheeled mobility devices since it is common for

a wheeled mobility device to encounter narrow passages in real situations. For instance, Figure 5 exhibits a T-Shape space in which a wheeled mobility device needs to make a 180 degree turn. Under the constraints of the given space dimensions, there are only two possible routes that a wheeled mobility device 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 wheeled mobility device must follow the same sequence of actions—going forward, going backward, and then going forward; there is no other alternative. Thus, the search space for this example can be conceptualized as three local configuration spaces (or C-spaces) connected via two narrow passages (see Figure 10). The motion planner generates a C-space from the geometric properties of the mobile device, the obstacles and workspace and attempts to construct the path in this configuration space. It is worthnoting that the kinematic constraints of the wheeled mobility device contribute the most to these narrow passages in addition to the geometrical constraints of the spatial dimensions.

Based on our experiments, 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 (wasted) efforts focus on sampling from other local spaces.

As has been shown in Figure 10, the narrow passages are located at the places where a wheeled mobility device performs a motion switch – changing its direction from forward to backward, or vice versa. Therefore, we observe that if a planner has control over its motion switch (instead of depending on randomness), it allows the RRT trees to approach the entries of a narrow passage before it takes samples from other local spaces,. Thus, 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 the narrow passage problem more efficiently.

2.4 The Enhanced RRT-ExtExt Planner

If we consider a wheeled mobility device to be a car-like robot maneuvering in a 2-D space, then its kinematic constraints can be represented as a list of input vectors applied to a set of state transition equations that indicate how the state of the wheeled mobility device changes over time. This section first discusses how to control the motions of a wheeled mobility device by manipulating the input vectors; we then present an enhanced RRT-ExtExt algorithm for solving the narrow passage problem.

2.4.1 State Transition Equations and Input Vectors

A wheeled mobility device 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 be represented mathematically using the notions of state transition equations and input vectors. Let each state of a wheeled mobility device be represented as **(x, y, ɵ)**, where **(x, y)** denotes the coordinates of the wheeled mobility device in a 2D space, **ɵ** 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 wheeled mobility device changes over time, given the current state and the current input (e.g., the steering angle of the wheeled mobility device). We use the following formulation of state transition equations:

$$dx = s*\cos ɵ$$

$$dy = s*\sin ɵ$$

$$dɵ = s/L*\tan ϕ$$

The speed of a wheeled mobility device 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 11. For example, a set of input vectors for a wheeled mobility device:

```
2   1.0   0.0      // going straight forward
2   1.0   0.4      // making a right turn while going forward
2   1.0   -0.4     // making a left turn while going forward
```

In the case that the turning radius of a wheeled mobility device is known, instead of the steering angle, we can compute the corresponding steering angle using the following formula:

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

where **R** denotes the turning radius of a wheeled mobility device. The derivation of the formula is fairly straightforward and is based on the geometrical relationship depicted in Figure 12.

In the original RRT-ExtExt, if a car-like robot can move both forward and backward, then the corresponding input vectors would include both positive speed and negative speed (a positive speed enables the robot to move forward, while a negative speed enables the robot to move backward). Users have no control over how these vectors are being chosen to grow an RRT tree in running time. Therefore, the chance that the planner can find a path going through narrow passages demonstrated in Section 2.3 is rather small.

The basic ideas of the enhanced RRT-ExtExt are as follows:

1. Instead of using one wheeled mobility device robot that maneuvers both forward and backward to expand an RRT tree, the new planner utilizes two wheeled mobility device robots, which have the

same geometric constraints, but different kinematic constraints (i.e., one goes forward only, and the other goes backward only);

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.

2.4.2 The Planning Algorithm

The enhanced RRT-ExtExt starts two wheeled mobility device robots to determine one path—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 wheeled mobility device motion planning problems can be solved by this setting (see Figure 13).

However, if a path has not been found 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. It then continues to search in the local C-space created by backward-only constraints. 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 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 wheeled

mobility device path planning problems can be solved quickly by using this approach, such as a bathroom configuration that requires one motion switch (see Figure 14).

The planning algorithm is described in Figure 15. In the algorithm, **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 wheeled mobility device motion planning problems.

3. IMPLEMENTATION IN A CAD ENVIRONMNENT

3.1 Background

Delivering practical CAD-based accessibility analysis tools has been a strong objective in our work. Previously, we describe an online client-server-based code-checking framework (Han 1998). Figure 16 illustrates a comparison between the traditional manual checking process and an automated online review process. The process of analysis using this framework is as follows:

1. The design data is created in a specialized AutoCAD environment.

2. An AutoLisp program consumes the design data to create an International Alliance for Interoperability (IAI) Industry Foundation Class (IFC) file (IAI 1997).

3. The AutoLisp program then launches a client program that sends the IFC file to the server which in turn analyzes the data and generates graphical and textual comments that are hyperlinked to each other and to the ADAAG guideline (similar to the web page generated by the hybrid framework presented above).

3.2 The Current Framework

For this research, we have focused on integrating the motion planner directly into Autodesk Architectural Desktop (ADT) leveraging internal ADT building model and foregoing the generation of IFC data and the transfer of this data from a client to a server. We have developed a computer tool that integrates ADT and the enhanced RRT-ExtExt algorithm for assessing wheeled mobility device accessibility of an architectural space. The tool shown in Figure 17 works as follows:

1. An AutoLisp program derives the geometric information from:
    a. the start and goal points of the accessible route as specified by the user
    b. the building model objects in ADT (for example, the walls generate obstacles)
2. The AutoLisp program creates a data file of geometric data in a format that the motion planner understands.
3. The AutoLisp program executes the motion planner (an executable program external to ADT) using the parametric data files that describe the wheeled mobility device and a set of input vectors that represent nonholonomic constraints of the wheeled mobility device.
4. The planner determines whether or not the space is accessible and sends results back to ADT.
5. If there is an accessible route, the user can delineate the route from a generated script with a sequence of the wheeled mobility device BLOCK.

4. RESULTS

This section discusses the possible usages of the computer tool developed. Specifically, we provide examples to illustrate the use of the computer tool for the evaluation of accessibility guidelines. Furthermore, we demonstrate how the tool can potentially be used to evaluate "barrier-free" design.

4.1 Evaluating Accessibility Guidelines

The power of the developed simulation-based tool is its ability to adjust the dimensions of both the wheeled mobility device and its turning parameters via a set of configuration files. This versatility is useful as it requires no recompilation of the tool to evaluate different guidelines or codes.

4.1.1 ADAAG

In the following, we illustrate the application of the motion planner to simulate three specific requirements as given in the ADAAG (1997). For all three examples, the wheeled mobility device assumes the measurements of the idealized wheelchair prescribed in the ADAAG.

The ADAAG requirement for a wheeled mobility device to make a 180 degree turn in a T-shape space is stated as follows:

> 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 (Access 1997).

After a corresponding case has been built using ADT, the result, a wheeled mobility device path, establishes the validity of the requirements described by the code (see Figure 18).

For the requirements of a 90-degree turn, the corresponding ADAAG provision states:

> 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 (Access 1997).

According to the developed computer tool, this particular configuration is indeed usable (see Figure 19).

The developed computer tool is also helpful in identifying what the turning parameters must be in order to satisfy a given provision. For example, in order for a wheeled mobility device to make a smooth U-turn, the ADAAG states the requirement of a space to be the following:

> The space needed for a smooth U-turn in a wheeled mobility device is 78 inches (1965 mm) minimum by 60 inches (1525 mm) minimum (Access 1997).

According to the results shown in Figure 20, a smooth U-turn is possible only if a wheeled mobility device has a turning radius of less than six inches. In order to make the turn, one wheel of the wheeled mobility device must roll backward while the other rolls forward.

4.1.2 ISO 7176-5

The following examples verify the usability of configurations as put forth by ISO 7176-5 (ISO 1984). From the abstract:

> The methods for determining the overall dimensions (both ready for occupation and folded), and the other characteristics of wheelchairs (manual and electric) are specified. The minimum turning radius and turn-around width are illustrated (ISO 1986).

Note that while the ADAAG specifies dimensions of the wheelchair shown in all the figures, ISO goes further and specifies dimensions for four classes of wheeled mobility devices and the turning radius of the devices as shown in Figure 21 (Ziegler 2003). The following set of examples utilizes the wheeled mobility device dimensions and turning radii as inputs to the motion planner's configuration files. The developed computer tool validates the cases as shown in Figure 22 to Figure 25.

4.2 Assisting Barrier-Free Design

Another practical use of the developed computer tool is to assist architects in designing barrier-free designs. One possible scenario would be:

1. An architect develops a floor plan using ADT and wants to check if the design is indeed satisfied by the accessibility requirements for wheeled mobility devices;

2. The architect runs the developed computer tool on the floor plan specifying a pair of designated locations where are expected to be accessible.

3. Based on the results, the architect modifies the design until the corresponding requirements are satisfied

4. When necessary, repeat steps 2 and 3.

Figure 26 illustrates a wheeled mobility device access for a given residential design.

5. SUMMARY AND DISCUSSION

We have presented the development of an enhanced RRT-based motion planner that utilizes random sampling in local C-space plus systematically expanding an RRT tree crossing different C-spaces separated by narrow passages, applied it to wheeled mobility device motion planning, and integrated it into the Autodesk Architectural Desktop environment. As illustrated, the developed tool can be potentially be used by standards organizations to validate existing and develop new performance-based accessibility guidelines. Furthermore, the tool can potentially be used by designers to validate accessibility conditions of a building and its facilities. As shown in the illustrative examples, the dimensions and turning constraints of a wheeled mobility device can be adjusted via the motion planner's configuration files. Thus, manufacturers of mobility devices would be able to test virtual prototypes of proposed wheeled mobility devices either against the appropriate set of standards for compliance approval or, in the case of customization, a client could give the manufacturer floor plans to validate accessibility of a given facility.

In our current work, we plan to re-integrate the improved wheeled mobility device motion-planning algorithms back into the previously described hybrid prescriptive-/performance-based accessibility analysis framework. In terms of ease-of-use, having the accessibility-route-analysis tool integrated into ADT has greatly enhanced usability. Directly integrating the analysis tools (including the hybrid framework) as opposed to executing the framework as an program external to ADT will broaden the usability of the developed tool.

6. REFERENCES

Access Board (1997), *Americans with Disabilities Act Accessibility Guide.* U.S. Architectural and Transportation Barriers Compliance Board, Washington, D.C.

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

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

Han, C.S., Kunz, J., & Law, K.H. (1998), *Client/Server Framework for On-Line Building Code Checking*. ASCE J. Computing in Civil Engineering, 12, 4, 181-194.

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

International Alliance for Interoperability (IAI) (1997), *Industry Foundation Classes, Specifications.* Volumes 1-4, Washington D.C.

International Organization for Standardization (ISO) (1984), *Wheelchairs -- Part 5: Determination of overall dimensions, mass and turning space*. ISO 7176-5, Geneva.

Kuffner, J., Kagami, S., Nishiwaki, K., Inaba, M., & 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. (1998), *Rapidly-exploring Random Trees: A New Tool for Path Planning.* TR 98-11, Computer Science Dept., Iowa State University.

LaValle, S. & Kuffner, J. (2000), *Rapidly-Exploring Random Trees: Progress and Prospects.* The Algorithmic Perspective. Fourth Int'l Workshop on the Algorithmic Foundations of Robotics, Hanover, NH.

Liu, Y. & 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, 86-93.

Ziegler, J., *Working Area of Wheelchairs Details about Some Dimensions that are Specified in ISO 7176-5,* International Workshop on Space Requirements for Wheeled Mobility, Center for Inclusive Designa dn Environmental Access, SUNY at Buffalo, October, 2003.

FIGURE LEGENDS

Figure 1: A test case of a university facility analyzed using the hybrid framework (Han 2002). The accessible route to be analyzed is noted by an initial point (the entrance) and a goal point (a toilet).

Figure 2: Analysis results displayed using a web-browser environment using the hybrid framework (Han 2002). For clarity the violation is circled. On a computer screen, the violation is highlighted in red -- the component (the water closet), the space (the toilet facility), and thus, the whole floor.

Figure 3: Validation of the violation (Han 2002).

Figure 4: The generated path through the modified facility using the hybrid framework (Han 2002).

Figure 5: Turning in a T-shape space (Access 1997).

Figure 6: Rapid exploration by the RRT algorithm (LaValle 1998) The algorithm rapidly explores the uncovered region before uniformly spreading over the entire space.

Figure 7: The RRT-ExtExt algorithm (LaValle 2000).

Figure 8: Examples of solution for car-like robotic nonholonomic problems using the RRT-ExtExt algorithm (LaValle 2000).

Figure 9: The narrow passage problem and the RRT trees.

Figure 10: The search space of a wheeled mobility device in a T-shape space (as shown in Figure 5)

Figure 11: Format of an input vector.

Figure 12: Geometrical relationship between the turning radius and the steering angle of a wheeled mobility device; (A – current state; B – new state; R – turning radius; s – speed; $d\square$ - the velocity of facing direction, and $d\square = s/L * \tan \phi$).

FIGURES

Figure 1

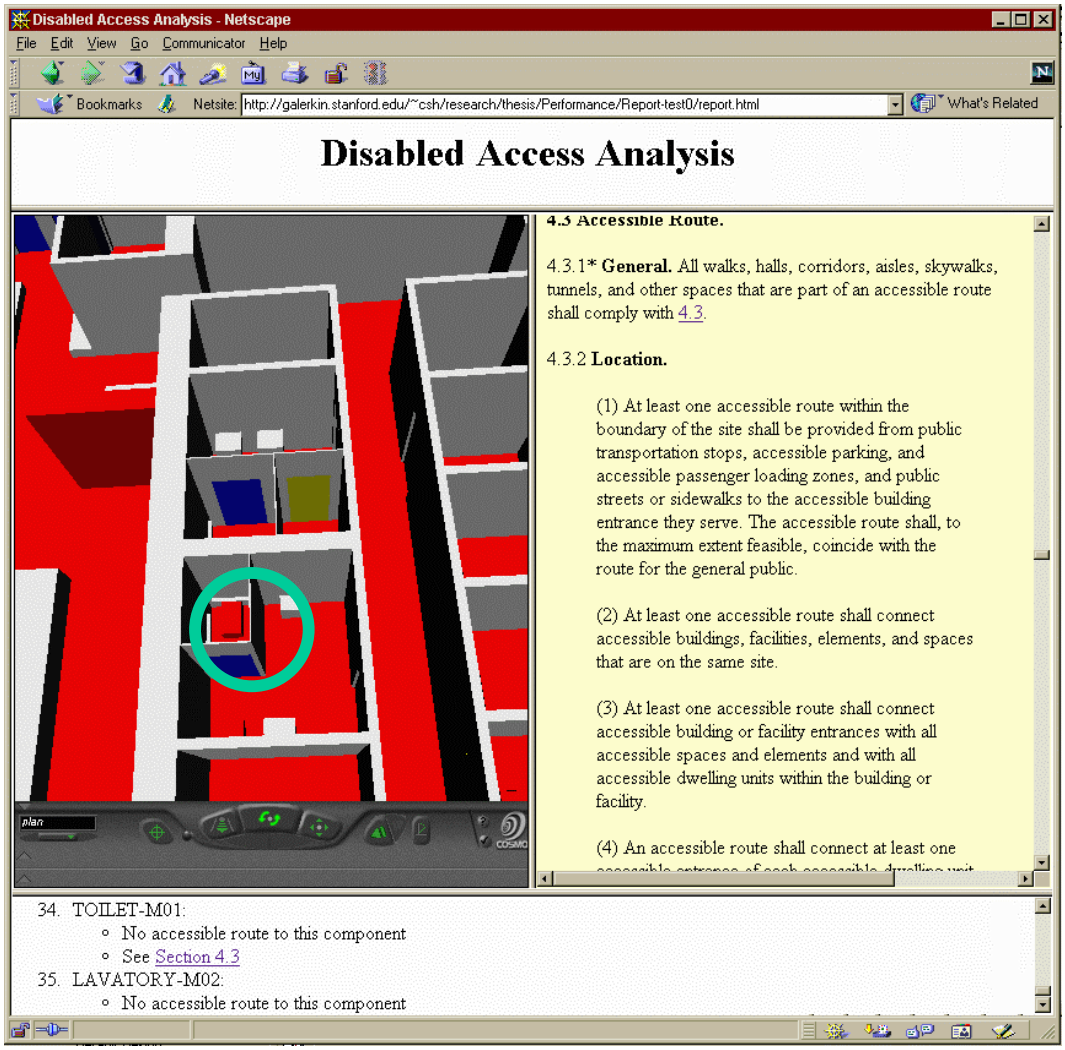

Disabled Access Analysis - Netscape
File Edit View Go Communicator Help
Bookmarks Netsite: http://galerkin.stanford.edu/~csh/research/thesis/Performance/Report-test0/report.html What's Related
Disabled Access Analysis
4.3 Accessible Route.
4.3.1* General. All walks, halls, corridors, aisles, skywalks, tunnels, and other spaces that are part of an accessible route shall comply with 4.3.
4.3.2 Location.
(1) At least one accessible route within the boundary of the site shall be provided from public transportation stops, accessible parking, and accessible passenger loading zones, and public streets or sidewalks to the accessible building entrance they serve. The accessible route shall, to the maximum extent feasible, coincide with the route for the general public.
(2) At least one accessible route shall connect accessible buildings, facilities, elements, and spaces that are on the same site.
(3) At least one accessible route shall connect accessible building or facility entrances with all accessible spaces and elements and with all accessible dwelling units within the building or facility.
(4) An accessible route shall connect at least one
34. TOILET-M01:
No accessible route to this component
See Section 4.3
35. LAVATORY-M02:
No accessible route to this component

Figure 2

Figure 3

Figure 4

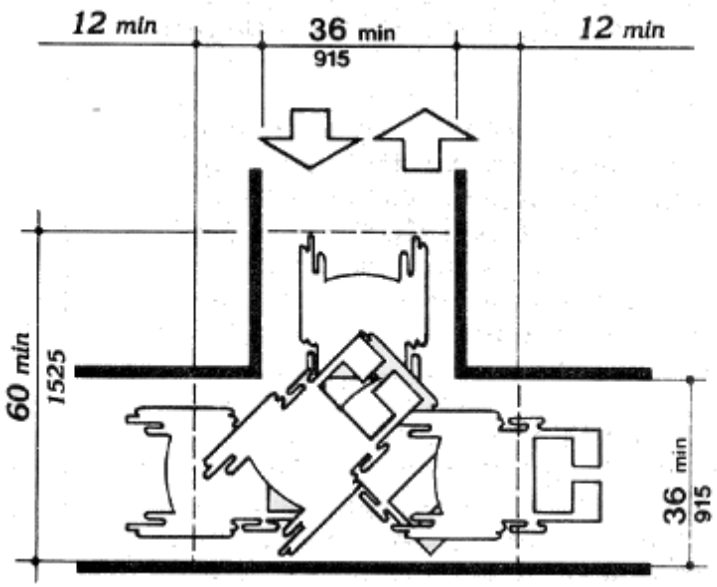


Figure 5

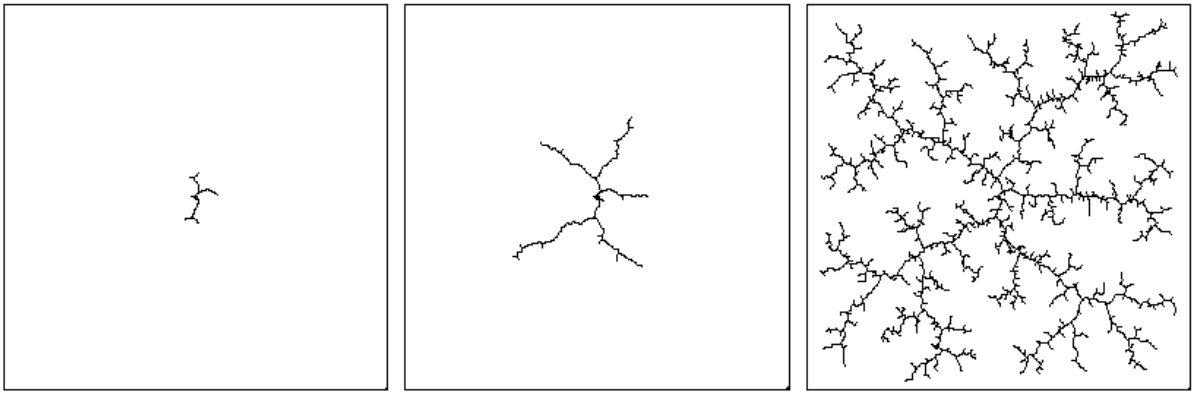


Figure 6

```
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 7

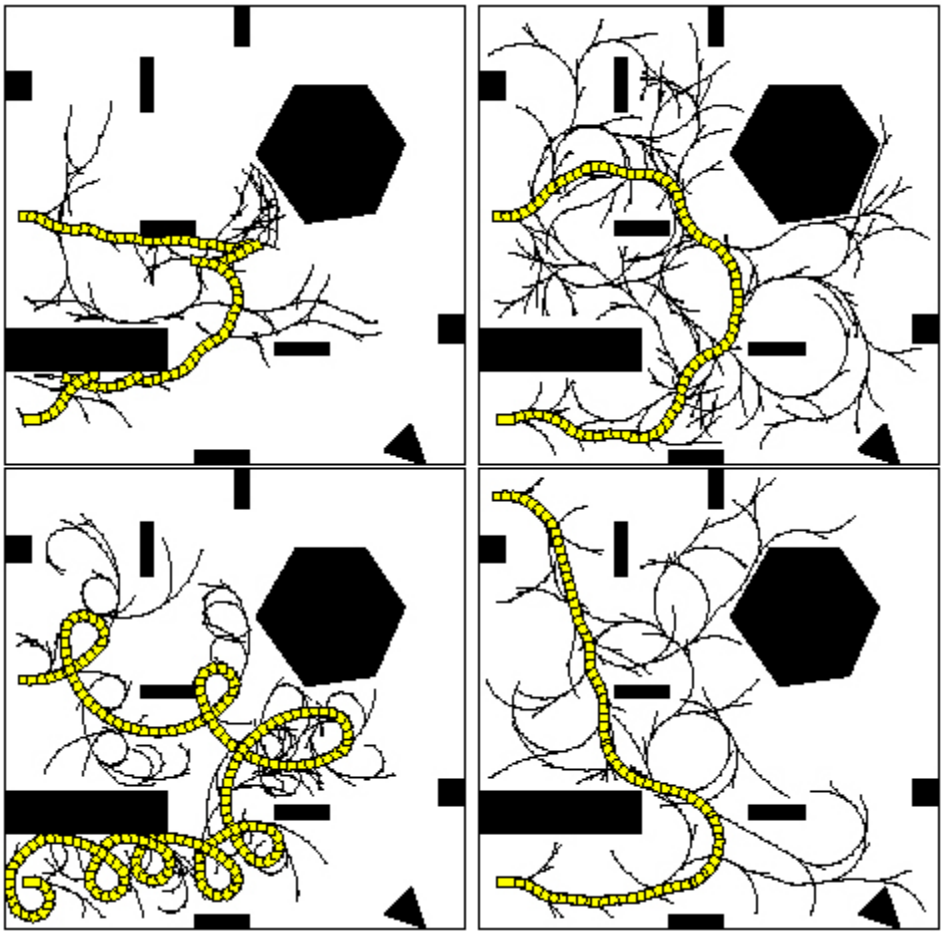


Figure 8

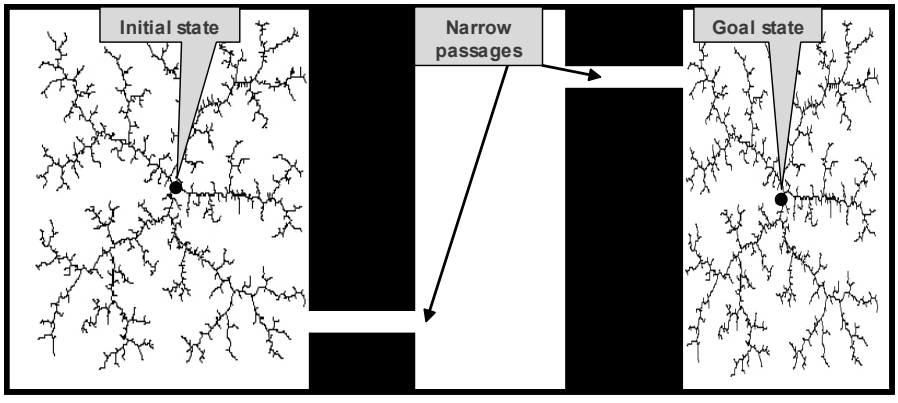


Figure 9

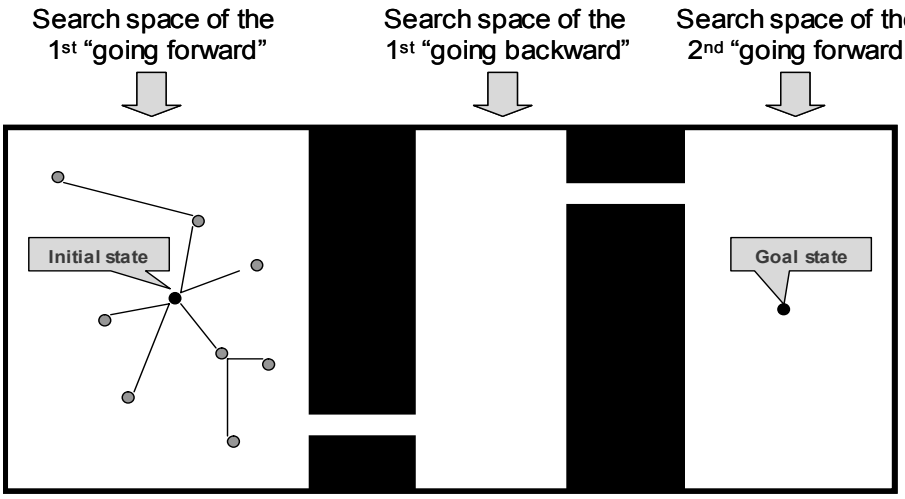


Figure 10

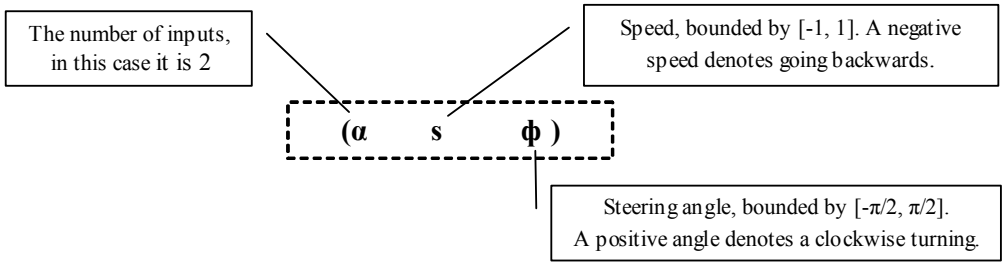


Figure 11

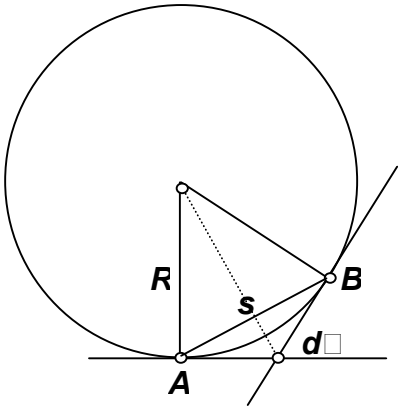


Figure 12

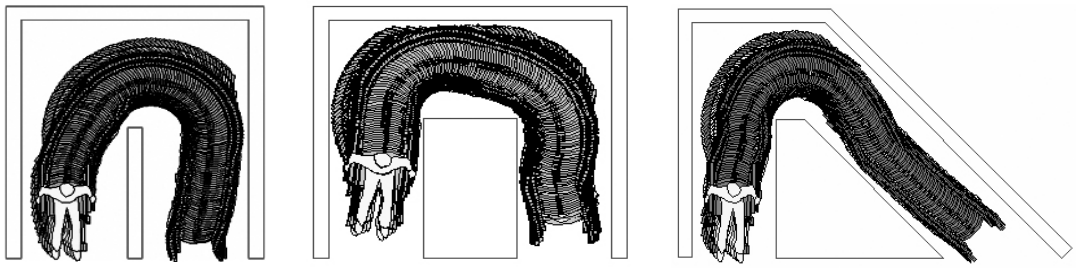


Figure 13

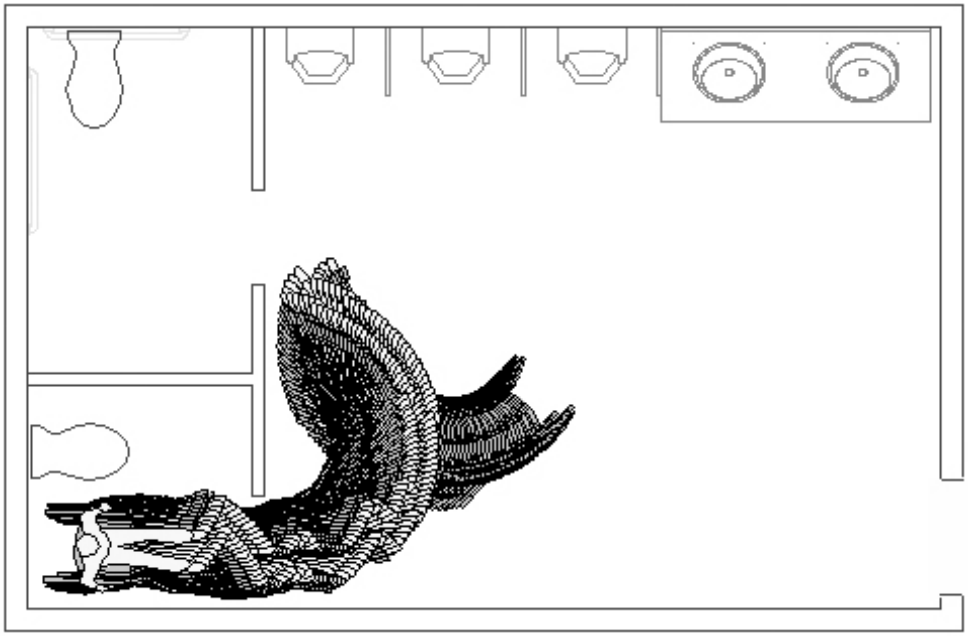
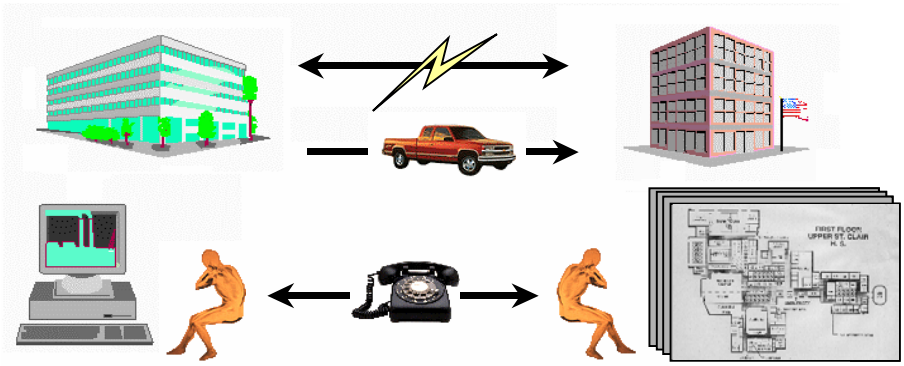


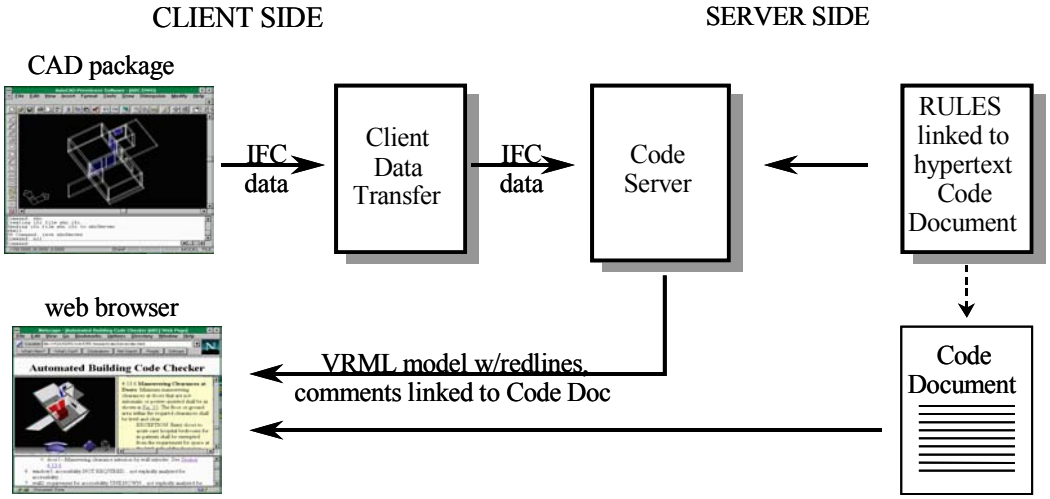
Figure 14

```
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 15



(a) Current manual and on-line review -- manual interpretation of design and code compliance



(b) On-line code checking -- IFC project model and automated code compliance checking

Figure 16

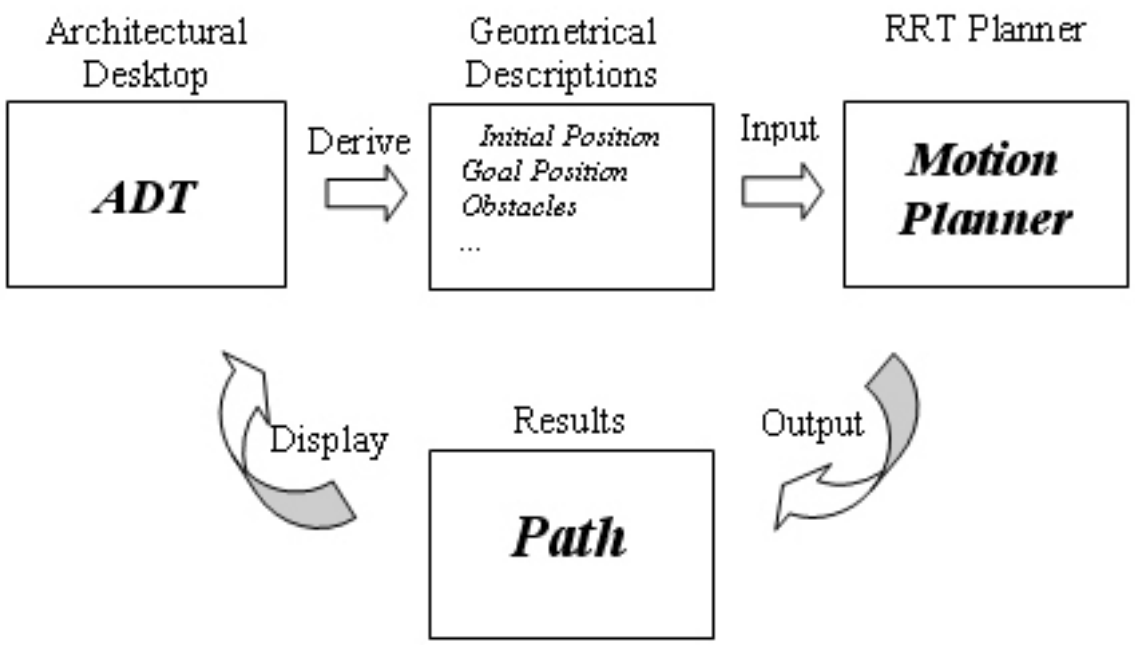

Architectural Desktop
ADT
Derive
Geometrical Descriptions
Initial Position
Goal Position
Obstacles
...
Input
RRT Planner
Motion Planner
Display
Results
Path
Output

Figure 17

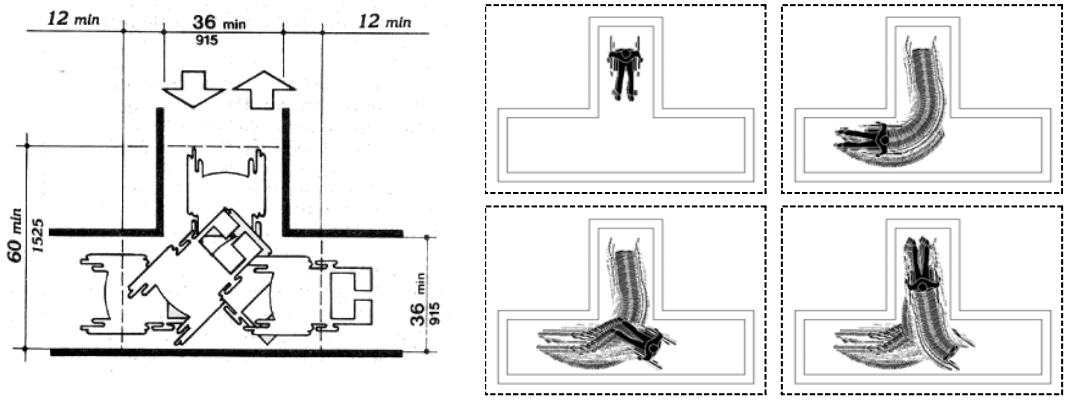


Figure 18

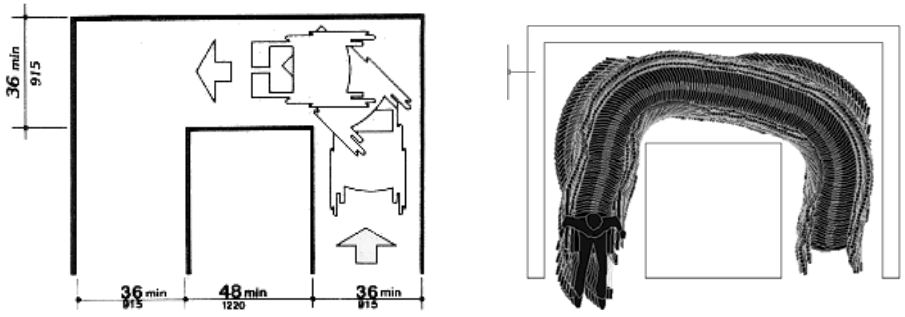


Figure 19

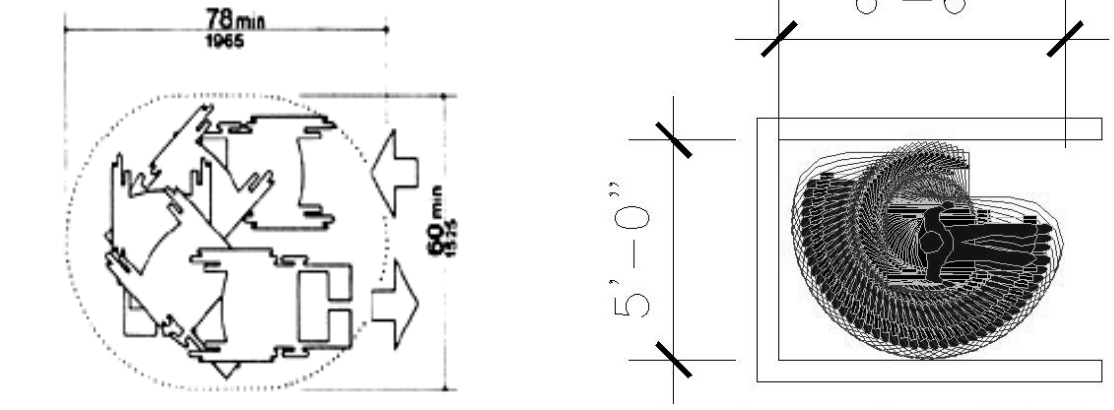


Fig. A2
Space Needed for Smooth U-Turn in a Wheelchair

Figure 20

Recommended maximum limits - manual wheelchair Occupied length : 1300mm -> 51.181" (p2) Occupied width : 800mm -> 31.496" (p2) Turning diameter: 2000mm -> 78.49" (p9) Turning radius : 0"	**Recommended maximum limits - electrically powered wheelchair - class A** Occupied length : 1300mm -> 51.181" (p2) Occupied width : 700mm -> 27.559" (p2) Turning diameter: 2000mm -> 78.740" (p9) Turning radius : 0”
Recommended maximum limits - electrically powered wheelchair - class B Occupied length : 1300mm -> 51.181" (p2) Occupied width : 700mm -> 27.559" (p2) Turning diameter: 2300mm -> 90.551" (p9) Turning radius : 18.948" (approximated)	**Recommended maximum limits - electrically powered wheelchair - class C** Occupied length : 1300mm -> 51.181" (p2) Occupied width : 700mm -> 27.559" (p2) Turning diameter: 2800mm -> 110.2359" (p9) Turning radius : 33.4216" (approximated)

Figure 21

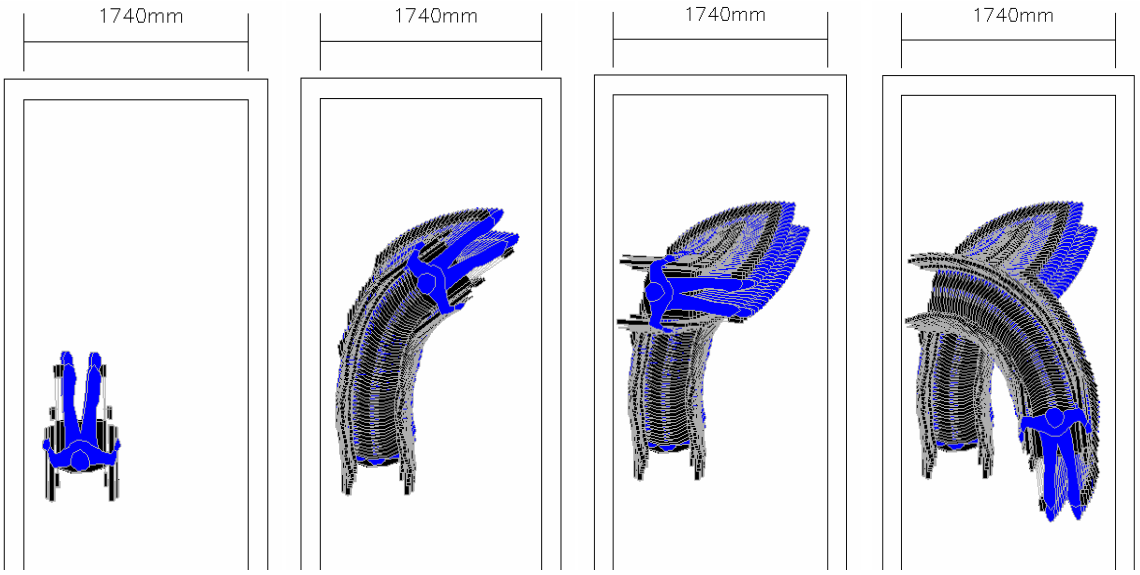


Figure 22

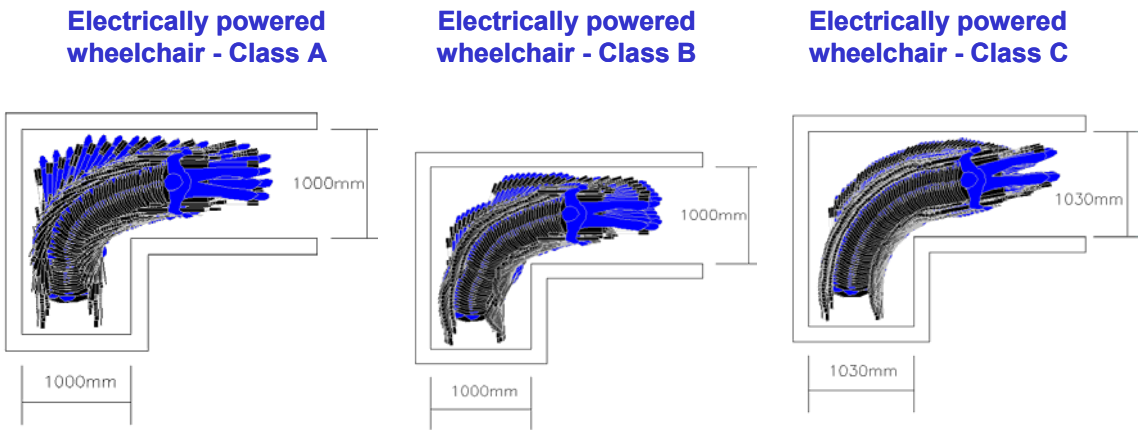


Figure 23

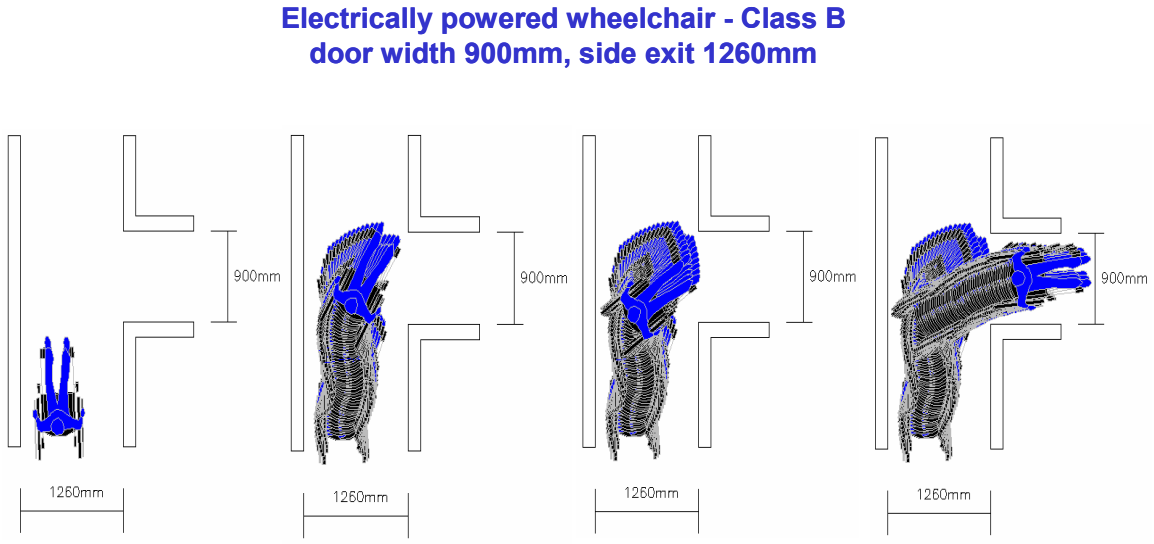


Figure 24

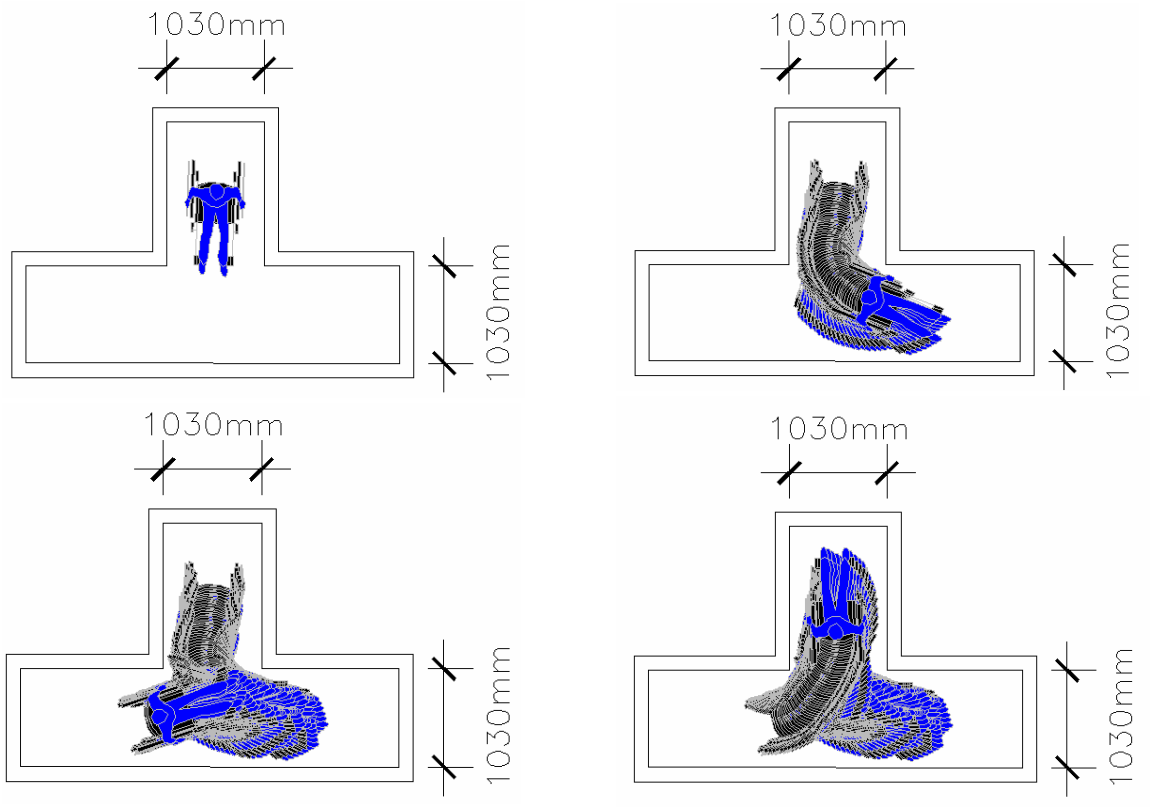


Figure 25

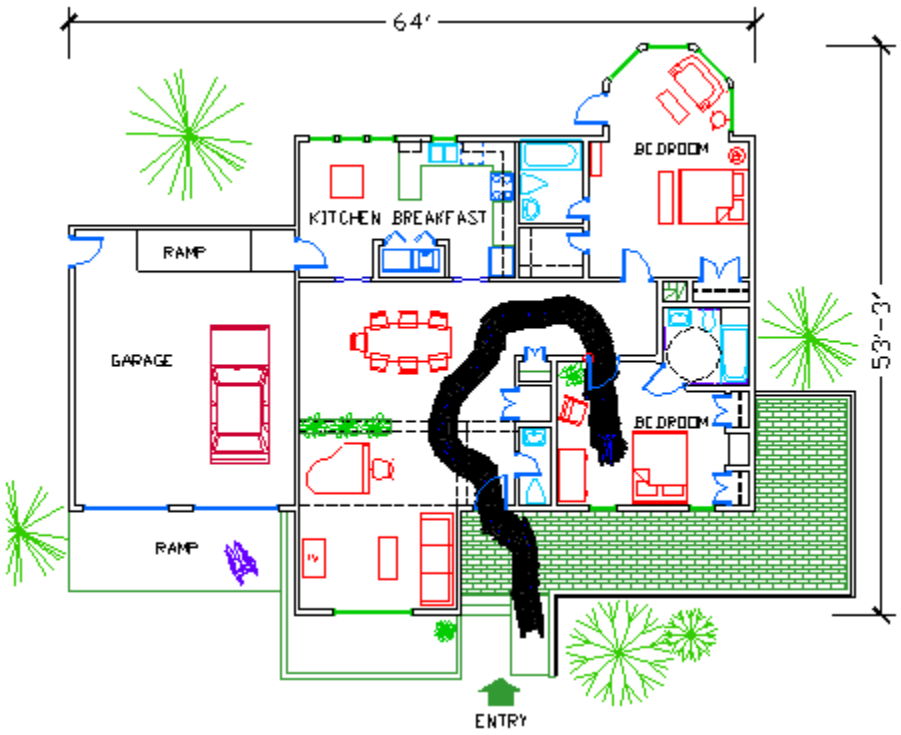

64'
53'-3"
BEDROOM
KITCHEN
BREAKFAST
RAMP
GARAGE
BEDROOM
RAMP
ENTRY

Figure 26