Time Efficient Hybrid Motion Planning Algorithm for HOAP-2 Humanoid Robot

Mohammed Elmogy, Christopher Habel, and Jianwei Zhang
Department of Informatics, MIN Faculty, University of Hamburg, Germany

Abstract

The development of practical motion planning algorithms and obstacle avoidance techniques is considered as one of the most important fields of study in the task of building autonomous or semiautonomous robot systems. The motion planners designed for humanoid robots combine both path planning generation and the ability of executing the resulting path with respect to their characteristics. These planners should consider the specific dynamical constraints and stability problems of the humanoid robots. In this paper, we present a time-efficient hybrid motion planning system for a Fujitsu HOAP-2 humanoid robot in indoor and miniature city environments. The proposed technique is a combination of sampling-based planner and D* Lite search to generate dynamic footstep placements in unknown environments. It generates the search space depending on non-uniform sampling of the free configuration space to direct the computational resources to troubled and difficult regions. D* Lite search is then implemented to find dynamic and low-cost footstep placements within the resulting configuration space. The proposed hybrid algorithm reduces the searching time and produces a smoother path for the humanoid robot with low cost.

1 Introduction

The motion planning problem has been studied for several decades and there are many algorithms that have been described in the literature [1, 2]. It is characterized by the ability of computing a collision-free feasible path for a mobile robot from a given initial position to a destination position through a workspace populated with stationary or moving obstacles. In some applications, the motion planning problem can be defined to maintain a set of constraints in the state of the world such as following a target, achieving knowledge about the world, or exploration in an unknown environment. Therefore, there are many different aspects to the motion planning problem, such as optimal path planning among rectangular obstacles, optimal path finding among weighted regions, and path planning to traverse narrow passages [3].

In the '80s and part of the '90s, finding collision-free paths was the main or only goal of the motion planning problem. Today, while obstacle avoidance remains a key issue, other important constraints are considered as well such as visibility, coverage, kinodynamic, optimality, equilibrium, and uncertainty constraints [1]. These constraints make motion planning problems more interesting and entail the implementation of more useful algorithms for real mobile robots.

In the last ten years, the significant progress in stable dynamic bipedal walking has been leading to an increased research interest in developing autonomous navigation strategies tailored specifically to humanoid robots. As autonomous navigation becomes an increasingly important research topic for humanoid robots, efficient approaches to perception, mapping, and motion planning, which are suited to their unique characteristics, will be required to integrate them easily in their typical operating environments. The ability of legged robots to step not only around but also over and onto some obstacles makes them particularly well suited for environments designed for humans, which often contain a wide variety of objects and obstacles such as furniture, stairs, doors, and uneven ground [4]. In addition to the bipedal walking of the humanoid robots, the large amount of their degrees of freedom (DOFs) should be considered while developing practical motion planning techniques for them. Typically, humanoid robots have 20 or more DOFs which must be controlled very carefully in order to maintain overall static and dynamic stability. These constraints severely restrict the set of allowable configurations and prohibit the direct application of existing motion planning techniques [5]. Motion planning techniques for humanoid robots should pose these particular challenges during the design phase.

2 Related Work

There exists an extensive literature on the motion planning problem in 2D static environments. Previous research with wheeled robots usually modeled a robot as a 2D circle and planned a path on a 2D grid map. These robots use a laser range sensor or a stereo vision sensor to generate a 2D map for planning. For example, Jan et al. [6] used a single circle and multiple circle modes for the robot to solve the narrow passages piano mover's problem in a 2D simulated

environment. In this approach, the single cell object travels along the optimal path in the grid plane without any collision. For the multiple circles model, the checkpoints of the single cell object are ensured to be collision free regarding all of the obstacles.

On the other hand, some attempts are reported in 3D motion planning for simulated and real humanoid robots. For example, Lau and Kuffner [7] presented a behavior planning approach to automatically generate realistic motions for animated characters. Motion clips are abstracted as high-level behaviors and associated with a behavior finite-state machine (FSM) that defines the movement capabilities of a virtual character. Shiller et al. [8] implemented a practical motion planner for animated human figures that can be also used for humanoid robots. They focused on path planning and sensor-based recognition of the environment. The human motions are modeled as a sum of rigid body and cyclic motions. They identified body postures that represent the rigid-body part of typical motion patterns. This leads to a model of the configuration space that consists of a multi-layered grid, each layer corresponding to a single posture.

Numerous research groups worldwide concentrate on the design and implementation of various motion planning algorithms that consider the bipedal capabilities of humanoid robots. For example, Bourgeot et al. [9] proposed a method to generate footprints from a reference path. This method finds a path and footprints in terrain under robot stability and motion continuity between starting and goal positions. The path is planned in a 3D simulated environment. The researchers studied the biped walking problem on horizontal flat and sloping grounds. Lorch et al. [10] have developed a sensor-based planning system using a biped robot with a stereo vision sensor. They proposed a footprints planner using a local environment map based on visual sensor inputs. This planner finds the step sequence while the robot is walking in a straight line. They recognized an obstacle under the assumption that any object in the scene is a rectangle. Okada et al. [11] described the vision-based navigation system for humanoid robots with vision-based floor recognition and path planning using a multi-layered body image. They utilized an RRT-Connect Planner as a path planning method. Kuffner et al. [5] developed a footprints and leg trajectory planner for humanoid robots using a global method. This method enables the robot to step over obstacles and consider global information to cope with local minima. Their approach adapts a variation of the randomized planner to compute full-body motions for humanoid robots that are both dynamically stable and collision-free. They employed a randomized search strategy based on Rapidly-exploring Random Trees (RRTs) [12]. Their algorithm has some limitations. First, their current implementation of the planner can only handle a fixed position for either one or both feet. Second, the effectiveness of different configuration space distance metrics needs to be investigated. Finally, they currently have no method for integrating visual or tactile feedback.

We developed a motion planning system for a humanoid robot to execute route based navigation tasks. The proposed motion planner operates at the level of footsteps and ignores the lower-level details of leg movements and control. It is used to solve the motion planning problem and handle the kinodynamic constraints of the HOAP-2 humanoid robot [13]. The planner is processed as two sequential phases. First, a sampling-based algorithm computes a collision-free path for the described route by ignoring system dynamics of the humanoid robot. Then, both the footstep planner and the motion trajectory generator are used to compute appropriate controls to implement the desired path and generate feasible motions for the humanoid robot.

3 HOAP-2 Motion Planner

The proposed planner is implemented to plan the motion and footstep placements for the humanoid robot while moving in the route environment. As shown in **Figure 1**, the motion planner is based on two main inputs. The first input is the initial path estimation which results from the symbolic representation of the processed route [14]. It provides the motion planner with the relationships between locomotion actions and landmarks. The second input is the processed vision data from the stereo vision and landmark detection stage of our humanoid robot navigation system [15]. It contains the information about the detected landmarks and their positions in the route environment.

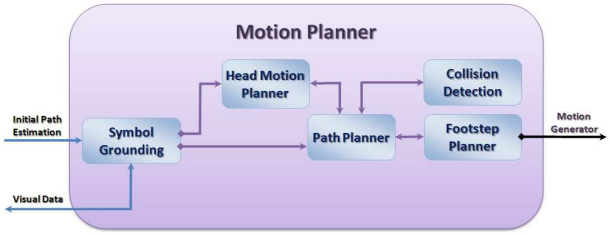

Figure 1: The architecture of the HOAP-2 humanoid robot motion planner.

The motion planner consists of five components: symbol grounding, head-motion planner, collision avoidance, path planner, and footstep planner. The symbol grounding phase is used to connect the symbolic representations of landmarks and actions to their equivalent perceptual landmarks and motion procedures, respectively. The output of the symbol grounding stage is provided to the head-motion planner to plan the motion of the robot's head with respect to the positions of the landmarks. It is also supplied to the path planner to generate the shortest feasible roadmap graph of the robot's path.

The path planner is implemented to extract the minimal feasible C_{free} and generate the shortest path to the target position. We used the Lazy Probabilistic RoadMap

mechanism (Lazy-PRM) with a non-uniform sampling to avoid the computational complexity of generating a denser search area. For collision detection, a cylinder model is used to approximate the trajectory for the body center of the humanoid robot during navigation. It calculates the actual areas required to execute different motion actions and compare them with the distances to the nearest obstacles.
Finally, the footstep planner is implemented to find smooth and low-cost footstep placements of the humanoid robot within the resulting C_{free}. It uses D* Lite search to reduce searching time and produces a smoother dynamic path for the humanoid robot at a low cost. In the following subsections, the building blocks of the motion planner will be discussed in detail.

3.1 Symbol Grounding

The symbol grounding stage is used to incorporate the high-level cognitive processes with their corresponding sensorimotor processes. The high-level cognitive processes perform abstract reasoning and generate plans for robot actions from the processed route. They use symbols to denote both landmarks and robot actions. Otherwise, the sensorimotor processes observe the physical world and execute actions in the route environment. If the overall system is to perform its tasks successfully, it needs to make sure that these processes are successfully connected to indicate the same physical objects.

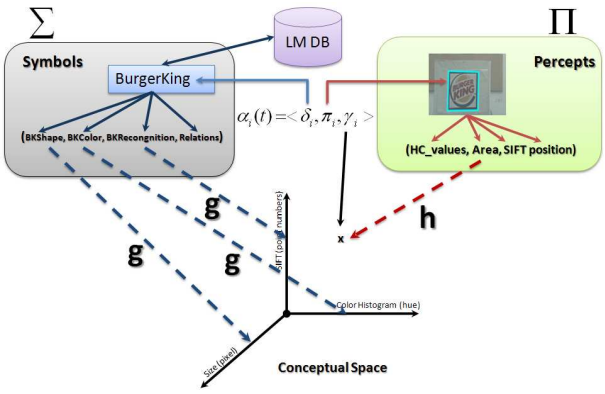

Figure 2: The anchoring process for a landmark between its symbolic and perceptual data.

To solve the symbol grounding problem, a methodology is needed to resolve situations where the sensors detect several landmarks that are consistent with the symbolic description of a desired landmark. In order to handle ambiguous situations, it needs to reason and act in a way that allows it to distinguish between the perceived objects and determine their correct correspondents. The plan involves finding out relevant information about the landmarks until the correct landmark is identified. We used the perceptual anchoring via conceptual spaces to connect the symbolic cognitive system (Σ) with its corresponding sensorimotor perceptual system (Π). **Figure 2** shows the anchoring process of the "BurgerKing" landmark that connects its symbolic representation with its perceptual region in the captured image. The conceptual space is a metric space whose dimensions, called qualities [16], are related with the quantities processed by the robot sensors. Points in a conceptual space, called knoxels [17], represent the epistemologically primitive elements at the considered level of analysis. For logo landmarks (such as BurgerKing), the conceptual space represents three different quantities: the correlation values to the hue component of the stored color histograms, landmark shape and size range, and the number of matched points resulting from the SIFT technique.
On the one hand, the symbolic system manipulates individual symbols for each landmark to denote its physical object. The predicate grounding function (g) associates each individual symbol with a set of symbolic predicates that assert properties of the corresponding landmark. It associates unary predicates in Σ to areas in the conceptual space. On the other hand, the perceptual system generates percepts from the observation of physical landmarks that are represented as regions in the captured images. The sensor model function (h) associates each percept with its observed values of a set of measurable attributes. It transforms a measurement vector from the sensor system into a set of knoxels in the conceptual space.
Therefore, the correspondence between symbols and percepts is reified in a data structure called anchor ($\alpha(t)$) that contains pointers to the corresponding symbols (σ_i) and percepts (π_i). In addition to these pointers, it has a pointer to an estimate of the current values of some attributes of the landmark which it refers to. This pointer is called the signature and denoted by γ_i which indicates its corresponding knoxels in the conceptual space. An anchor can be considered as a model of a physical object that reflects the persistence of the object, and which can be shared across different subsystems of the agent. Once an anchor has been created, it should be continuously updated to account for changes in the landmark's attributes and handle the relation of this landmark with other landmarks in the route description. This connection is made depending on the relationships that are retrieved from the processed route to handle the uncertainty during robot navigation.
We extended the anchoring process to also handle robot actions. The anchoring process is used to connect the symbolic representation of the robot locomotion actions to their corresponding dynamic procedures which are controlled by path and footstep planners.

3.2 Head-motion Planner

In real and unknown environments, usually motion planners only focus on how to find an optimal path to the destination, but it is also important to decide on where to explore and look in order to accomplish finding a path to the goal. When the humanoid robot is following a path, its head should be moved according to the situation pre-

sented in the initial path estimation to detect and localize the landmarks. Therefore, the motion planner should generate walk and head-motion commands and send them to the motion trajectory generator to execute them.
Accordingly, we implemented the head-motion to plan the movement of the robot's head depending on the direction of landmarks in the estimated initial path and the movement ranges of the neck's motors. The robot will tilt its head to the right or left to detect the landmarks which are located at the road sides. It also looks down to the floor to detect landmarks such as crossroads and street boundaries in the miniature city.
The planner checks the direction of the landmark from the estimated path. Then, the robot changes its head direction depending on the location of the processed landmark. If the robot fails to detect the landmark, the planner changes its head direction by $15\,^{\circ}$. It processes again until it detects the landmark or terminates if the angle of the robot's head equals 0.

3.3 Collision Detection

Collision detection is considered to be one of the crucial factors in motion planning. For humanoid robots, there is an effective and simple way to detect collision by choosing an appropriate bounding volume approximating the shape of the robot. A trajectory for the body-center of a humanoid robot is computed by approximating its shape by using a cylinder surrounding its body. A cylinder model is useful during humanoid robot turns and lateral walking to calculate the actual processing space required to execute robot actions. As the positions of the nearest obstacles to the robot are calculated by using triangulation, a cylinder model of the humanoid robot can be checked for obstacle avoidance in a constant time. Simply, if the distance between the robot and the obstacle is known, then it will be compared to the radius of the cylinder model.

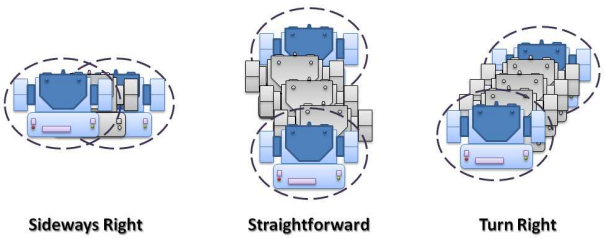

Figure 3: Different motion actions of the humanoid robot by using a cylinder model.

The cylinder is a tight fit to the shape of the humanoid robot standing still. When the robot moves, however, additional space is needed for the transition in C-space as body and legs are swinging while walking. **Figure 3** shows snapshots of the robot when moving forward, turning right and stepping sideways. We account for the additional space by enlarging the cylinder at the start and end configurations depending on the action which will be processed by the robot. When the robot walks sideways, no additional space is needed. If the robot walks straightforward, it needs additional space with respect to the swinging of its legs while walking. Therefore, the obstacle distance is compared to the enlarged cylinder radius plus the expected step distance. For turns, the humanoid robot wants extra turning space in a cylinder model, and then the cylinder model is enlarged by twice the turn radius to let the humanoid robot turn in a specific direction without collision. Therefore, such an approximation enables a humanoid robot to find paths in real-time and include actions such as walking sideways through a narrow space.

3.4 Path Planner

The path planner can be considered as the core stage of our motion planner. It generates the shortest roadmap graph (G) of the robot's path. It only returns the path graph, not the ability to execute that path. In our system, the path planner only depends on both the processed route description, as initial path estimation, and the retrieved data from vision [14, 15]. The path planner processes the path as segments from the generated topological map [18]. Each segment represents the distance between two adjacent landmarks in the robot's estimated path. The planner processes each segment as an independent path with its own start and end points. It is continuously evaluating the current C_{free} and sends the resulting roadmap graph of the processed path segment to a footstep planner which computes the next footsteps of the humanoid robot. The resulting footstep placements and robot actions are fed to the robot's motion trajectory generator to execute the motion of the robot.
As the sampling-based motion planning algorithms present practical and efficient solutions for the motion planning problem, they are extensively applied to many problems in high-dimensional configuration spaces. Therefore, we implemented our path planner by using a modified version of the PRM planner [2, 19]. We applied the lazy evaluation [1, 20] to the PRM planner with non-uniform sampling to handle narrow passages. The main theme of the planner is to minimize the number of collision-checks performed during planning. By avoiding local planning and instead keeping the global view, only the part of C-space that is essential in answering a query is explored.
The path planner minimizes the running time by reducing the number of collision checks performed during planning. It initially assumes that all nodes and edges in the roadmap are collision-free, and searches the roadmap for the shortest path between the start and goal nodes. The nodes and edges along the path are then checked for collision. If a collision with an obstacle occurs, the corresponding nodes and edges are removed from G. It updates the roadmap with new nodes and edges, and then searches for a shortest path. The above process is repeated until a collision-free path is returned. To avoid bad estimations for the path

planner, we used limited time for processing and generating G. **Figure 4** illustrates the algorithm of the proposed path planner.

At the beginning, a low resolution regular grid is applied to C-space to generate a search grid. The C-space is divided into small cells. Each cell has an associated location in the grid (x, y) and an information value. The grid generation will help in maintaining the connectivity of the graph by defining a minimum discretization for the open spaces. The discretization density is adjusted to suit the environment, selecting as sparse a grid as possible. Up to this stage, the cells hold only position information.

```
input  : n← the number of nodes in the roadmap
         k← the number of the closest neighbors to c
output: A roadmap G = (V, E)
         P ← a path from c_init to c_goal

foreach S_i ∈ Initial Path Estimation do
    V ← {}, E ← {};
    c_init ← the initial configuration of S_i;
    c_goal ← the goal configuration of S_i;
    V ← V ∪ {c_init};
    GridGeneration();
    repeat
        c ← a random configuration in C-space;
        if c ∈ C_obst then
            ć ← a random configuration in C-space;
            if ć ∈ C_obst then
                c_m ← the midpoint of cć line segment;
                if c_m ∈ C_free then V ← V ∪ {c_m};
        else
            V ← V ∪ {c};
    until |V| > n ;
    V ← V ∪ {c_goal};
    forall c ∈ V do
        N_c ← the k nodes of c from V;
        forall ć ∈ N_c do
            c_s ← closest neighbor of c from N_c in c_goal direction;
            if (c, c_s) ∉ E AND △(c, c_s) ≠ NIL then
                E ← E ∪ {(c, c_s)};
                break;
            else
                N_c ← N_c \ {c_s};
    P ← shortest path (c_init, c_goal, G);
    if P ∉ ∅ AND P ∈ C_free then
        return P;
    else if t < t_max then
        PathEnhancement();
    else
        return failure;
```

Figure 4: The proposed algorithm of the path planner.

The path planner minimizes the on-line computation by pre-generating a search space to contain all the information that will be used during the on-line path planning. It also avoids generating an unnecessarily complete and complex space. It samples the C_{free} by using a non-uniform approach returning with the minimal free search space to avoid the computational complexity of generating a denser search area. The C-space is sampled by using the bridge test approach [21] that was introduced to boost the sampling density inside narrow passages using only a simple test of the local geometry. The idea is to take two random samples, where the distance between the samples is chosen according to Gaussian distribution. Only if both samples lie in C_{obst} and the point in the middle of them lies in C_{free}, the free sample is added. Increasing the density of sampling around narrow passages increases the chances of finding samples in areas that are hard to reach and are likely to be needed for finding a solution. On the other hand, the samples in open space are randomly chosen in the medial of C_{free} with lower density.

The planner builds a roadmap graph of a feasible path by lazily evaluating the feasibility of the roadmap as planning queries are processed. The c_{init}, c_{goal}, and a number of non-uniformly distributed samples are used to form nodes in a roadmap. They are connected by edges in which each pair of nodes are sufficiently close together and in c_{goal} direction.

The second step in the algorithm is to find the shortest path in G between c_{init} and c_{goal}. Given a procedure that estimates the length of a path, the shortest feasible path in the roadmap is found by repeatedly searching for the shortest path by using D* Lite search. Therefore, the path is checked to know if it is collision-free or not. If the resulting path lies in C_{free}, it will be supplied with the grid of cells of the C-space to the footstep planner to retrieve the actual footstep placements for the humanoid robot.

On the other hand, if no path exists in the roadmap, the planner either reports failure or go to the path enhancement stage to add more nodes to the roadmap and start searching again. The choice is determined by the overall time allowed to solve the problem. If the planner reports an occurrence of a collision, the corresponding node or edge from the roadmap is removed. Then, the planner adds new nodes and edges and searches again for the new shortest path.

3.5 Footstep Planner

The footstep planner is a high-level planner implemented to calculate footstep placements under humanoid robot stability and motion constraints. It ignores as much of the underlying details of leg movement and trajectory generation as possible, and works instead from a description of the robot's capabilities. By using the roadmap graph as a reference path, it returns a sequence of footholds that can be carried by the robot to reach the goal location.

The footstep planner takes the roadmap graph, the initial and goal points, and the grid of feasible cells as inputs. It returns the solution as an ordered list of the footstep placements that should be executed to reach the goal position. The D* Lite search [22] is used to determine repeatedly the shortest paths between the current footstep of the humanoid robot and the goal location as the edge costs of a graph change while the robot moves towards the goal position. D* Lite search provides accurate and fast solutions for humanoid robot motion in dynamic and unknown environments. It generates the shortest and the lowest-cost sequence of footstep locations to reach the target point. D* Lite works by exploring grid nodes (cells) that are provided by the path planner and calculates the cost function $F(n)$

for each cell in the roadmap graph. The cost function is calculated as the sum of the following three costs:

1. **Step costs** ($G(n)$ **&** **rhs**(n)): They are the costs of making the desired step from the start node to the current node (n). The cost grid is an eight-connected grid whose edge costs are initially one. The value of the cell is changed to infinity when the robot discovers that this cell cannot be traversed.

2. **Estimated heuristic cost** $H(n)$: It is the estimated cost from the current node (n) to the goal node. It uses a heuristic search to estimate the cost of the goal node and it minimizes the cost of the path so far. D* Lite search is optimal if the estimated cost to the goal is always underestimated.

3. **Clearance cost** $C(n)$: It is used to insure that the generated footsteps are directed to the middle of C_{free} and they are not adjacent to the obstacles. It calculates the clearance cost of following G that is generated from the path planner.

After the cost functions have been calculated, the planner computes the optimal sequence of footstep locations to reach the desired goal. The robot actions are modeled by storing a symmetric collection of candidate footstep transitions for both feet. A sequence of footstep placements to reach a goal in the route environment is computed from a discrete set of feasible footstep locations corresponding to stable candidate stepping motion trajectories. The planner returns the solution as an ordered list of the footsteps which should be processed to reach the goal. To achieve smooth walking, the parameters of the next step must be known before the current step of the robot ends.
D* Lite search can efficiently recalculate the shortest path from the current position of the robot to the goal position. It only recalculates those goal distances that have been changed or have not been calculated before. It achieves a speed up of one to two orders of magnitude over repeated A* search by modifying previous search results locally.

4 Motion Trajectory Generator

After estimating the footstep placements and the head movements for the humanoid robot, they are submitted to the motion trajectory generator to execute the desired actions. The motion trajectory generator has three fundamental functions: (i) keeping the humanoid robot in balance, (ii) moving the swing leg, and (iii) controlling visual attention. The motion trajectory generator calculates the walking parameters and sends them to the robot's actuators to execute these actions.
We considered the walking process of the humanoid robot as a symmetric, periodic and smooth motion. The motion trajectory generator is used in order to output a final dynamically-stable trajectory. We use the ZMP [23] trajectory in order to maintain overall dynamic stability. The ZMP walking pattern is used to produce humanoid robot gaits as dynamic and stable as possible. The foot placement actions indicate the motion, turns, and change of orientation actions. These actions are divided into six footstep placement actions for the humanoid robot: straightforward, straight backward, turn right, turn left, sideways right, and sideways left. All of these actions have two parameters which have real values. **Figure 5** shows some foot placement actions and their parameters for the HOAP-2 humanoid robot. It is worth noting that the humanoid robot does not necessarily need to be able to exactly perform these six actions. For example, the robot could well use several footsteps for performing the 45° rotation in turn actions.

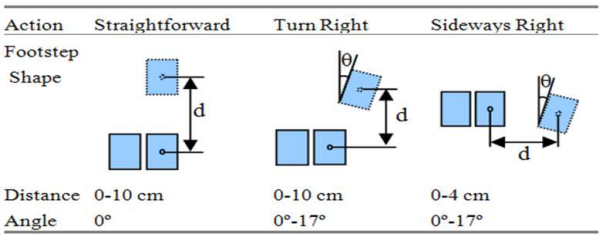

Figure 5: Footstep placements for HOAP-2 humanoid robot.

On the other hand, the robot's head is moved depending on the direction of the processed landmarks and the movement range of the neck's motors. The robot will tilt its head to the right and the left to detect the landmarks which are located at the road sides. It also looks down to the floor to detect landmarks such as crossroads and street boundaries in the miniature city. The head orientations of the humanoid robot are divided into four actions: turn right, turn left, move up, and move down.

5 Implementation

We have implemented our approach on the second generation of Fujitsu's Humanoid for Open Architecture Platform (HOAP-2) [13]. HOAP-2 is equipped with 25 servo actuators. It has four force sensing registers (FSRs) in each foot to detect reaction forces from the floor. It is also equipped with an accelerometer and gyroscope inside the torso. The vision system consists of two CMOS cameras, capable of capturing frames of 320X240 pixels at 25fps.
The navigation task is described by the user as route instructions via a graphical user interface (GUI). The navigation process is executed in a miniature city which is built on a 5m x 3.2m area. It contains buildings of recognizable shapes and colors (such as the railway station and the town hall) and other buildings of unique characteristics and symbols.
We tested the proposed algorithm on simple routes and it takes approximately 1.7 sec to process each segment in the

processed route. We are still working on handling long and more complex routes which contain narrow passages.

6 Conclusion

In this paper, we presented the anatomy of our proposed motion planning system for a Fujitsu HOAP-2 humanoid robot. The proposed technique is a combination of a sampling-based planner and a D* Lite search to generate fast and dynamic footstep placements for the humanoid robot in unknown environments. The robot navigation is based on the route described by the user to generate initial path estimation to the navigation task. The humanoid robot begins from the start point and moves along that path to collect information and recognize the landmarks by using its stereo vision and implemented techniques of landmark recognition. Based on the new findings and the processed route, the path is then re-planned to adjust the robot's position during navigation.

The main task of our motion planner is to find a sequence of actions as close to optimal as possible that causes the robot to reach its goal location while avoiding the obstacles. The path planner is implemented to extract the minimal feasible C_{free} and generate the shortest path to the target position. We used a modified version of the Lazy-PRM technique with a non-uniform sampling to avoid the computational complexity of generating a denser search area. The planner directs the computational resources to troubled and difficult regions, such as narrow passages, leaving the larger open spaces sparsely populated.

The footstep planner is implemented to find smooth and low-cost footstep placements of the humanoid robot within the resulting C_{free}. It uses D* Lite search to reduce searching time and produces a smoother dynamic path for the humanoid robot at a low cost.

References

[1] Choset, H.; Lynch, K. M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L. E.; Thrun, S.: *Principles of robot motion-theory, algorithms, and implementations*, MIT Press, 2005

[2] LaValle, S. M.: *Planning algorithms*, Cambridge University Press, 2006

[3] Tsianos, K. I.; Sucan, I. A.; Kavraki, L. E.: *Sampling-based robot motion planning: towards realistic applications*, Computer Science Review, vol. 1, no. 1, 2007, pp. 2-11

[4] Michel, P.; Chestnutt, J.; Kagami, S.; Nishiwaki, K.; Kuffner, J. J.; Kanade, T.: *Online environment reconstruction for biped navigation*, In Proceedings of the 2006 IEEE International Conference on Robotics and Automation(ICRA'06), Florida, 2006, pp. 3089-3094

[5] Kuffner, J. J.; Nishiwaki, K.; Kagami, S.; Inaba, M.; Inoue, H.: *Motion planning for humanoid robots under obstacle and dynamic balance constraints*, In Proceedings IEEE International Conference on Robotics and Automation(ICRA'01), Seoul, Korea, 2001, pp. 692-698

[6] Jan, G. E.; Chang, K. Y.; Parberry, I.: *Optimal path planning for mobile robot navigation*, IEEE/ASME Transactions on Mechatronics, vol. 13, no. 4, 2008, pp. 451-460

[7] Lau, M.; Kuffner, J: *Behavior planning for character animation*, In 2005 ACM Siggraph/Eurographics Symposium on Computer Animation, Los Angeles, CA, 2005, pp. 271-280

[8] Shiller, Z.; Yamane, K.; Nakamura, Y.: *Planning motion patterns of human figures using a multi-layered grid and the dynamics filter*, In Proceedings 2001 IEEE International Conference on Robotics and Automation (ICRA'01), 2001, pp. 1-8

[9] Bourgeot, J.; Cislo, N.; Espiau, B.: *Path-planning and tracking in a 3d complex environment for an anthropomorphic biped robot*, In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS'02), 2002, pp. 2509-2514

[10] Lorch, O.; Albert, A.; Denk, J.; Gerecke, M.; Cupec, R.; Seara, J. F.; Gerth, W.; Schmidt, G.: *Experiments in vision guided biped walking*, In Proceedings of the 2002 Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS'02), 2002, pp. 2484-2490

[11] Okada, K.; Inaba, M.; Inoue, H.: *Walking navigation system of humanoid robot using stereo vision based floor recognition and path planning with multi-layered body image*, In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03), Las Vegas, Nevada, 2003, pp. 2155-2160

[12] Ferguson, D.; Kalra, N.; Stentz, A.: *Replanning with RRTs*, In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'06), 2006, pp. 1243-1248

[13] Fujitsu Automation Co., Ltd.: *HOAP-2 Instruction Manual*, 2004

[14] Elmogy, M.; Habel, C.; Zhang, J.: *A cognitively motivated route-interface for mobile robot navigation*, *I*n proceedings of the 3rd International Workshop on Human-Centered Robotic Systems (HCRS'09), Bielefeld, Germany, 2009, pp. 73-82

[15] Elmogy M.; Zhang, J.: *Robust real-time landmark recognition for humanoid robot navigation*, In

proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics (ROBIO'08), Bangkok, Thailand, 2008, pp. 572-577

[16] Gärdenfors, P.: *Conceptual spaces: the geometry of thought*, MIT Press, 2000

[17] Chella, A.; Coradeschi, S.; Frixione, M.; Saffiotti, A.: *Perceptual anchoring via conceptual spaces*, In Proceedings of the AAAI-04 Workshop on Anchoring Symbols to Sensor Data, 2004

[18] Elmogy, M.; Habel, C.; Zhang, J.: *Robot topological map generation from formal route instructions*, In Proceedings of the 6th International Cognitive Robotics Workshop at 18th European Conference on Artificial Intelligence (ECAI'08), Patras, Greece, 2008, pp. 60-67

[19] Lindemann, S. R.; LaValle, S. M.: *Current issues in sampling-based motion planning*, Robotics research, 2005, pp. 36-54

[20] Bohlin, R.; Kavraki, L. E.: *Path planning using lazy PRM*, In IEEE International Conference on Robotics and Automation (ICRA2000), San Francisco, CA, USA, 2000, pp. 521-528

[21] Hsu, D.; Jiang, T.; Reif, J.; Sun, Z.: *The bridge test for sampling narrow passages with probabilistic roadmap planners*, In proceedings IEEE International Conference on Robotics and Automation (ICRA'03), 2003, pp. 4420-4426

[22] Koenig S.; Likhachev, M.: *D* Lite*, In Proceedings of the AAAI Conference of Artificial Intelligence, 2002

[23] Vukobratovic, M.; Borovac, B.: *Zero-moment point -thirty five years of its life*, International Journal of Humanoid Robotics, vol. 1, no. 1, 2004, pp. 154-173