Online Object Searching by a Humanoid Robot in an Unknown Environment

Masato Tsuru[1,2,3], Adrien Escande[1,2], Arnaud Tanguy[1,2], Kevin Chappellet[1,2], and Kensuke Harada[2,3]

***Abstract*—This paper proposes a framework for an autonomous humanoid robot, aimed at searching for a target object in an unknown environment using 3D-simultaneous localization and mapping (SLAM). The robot determines, *while walking*, the next viewpoint from an environment map and aggregated object recognition results, and automatically finds and grasps the target object. Whereas most robot exploration studies require a static map, hints regarding object position, area size limitations, or offline viewpoint planning time for each observation, our system can globally find an occluded object in an unknown environment, based only on the 3D target model. The biggest novelty of this research is that this framework always runs its viewpoint planner in background and immediately updates the destination if the camera gets environment/object information. To follow that goal change quickly, the humanoid robot re-plans its footstep trajectory without stopping, using foot landing estimation based on 3D-SLAM's localization. Notably, our robot can predict an unobserved area, and actively reveal it while avoiding obstacles. We validated the efficacy of this method through real experiments with an "HRP2-KAI" in several environments, and achieved fully automated searching and grasping.**

***Index Terms*—Humanoid Robot Systems, SLAM, Vision-Based Navigation, Perception-Action Coupling**

I. Introduction

HUMANOID robots are expected to act autonomously to accomplish a variety of tasks in a variety of industries experiencing labor shortages, such as in construction, nursing care, and domestic support. One of the typical tasks expected of such robots is to find, grasp, and bring objects that humans need. In this research, therefore, we set our goal to find and grasp a target object, typically a hand tool, in an unknown indoor environment. There are four typical technological elements required for a humanoid robot to autonomously search for and grasp a target object: object recognition, environment recognition, action planning, and bipedal walking. A significant amount of research has been conducted in recent years on these technologies. For example, the convolutional neural networks have allowed for excellent innovation in the field of "object recognition"[1][2]. In the field of "environmental recognition", research is being conducted on constructing surrounding environments using LiDAR and map-based route planning[3]. In the "action planning" field, a method for obtaining advanced action strategies based on sensor inputs has been proposed, based on incorporating deep reinforcement learning[4] and a partially observable Markov decision process (POMDP)[5]. In the field of "bipedal walking", research has been conducted on balance control[6] and footstep planning. Most of these studies have been conducted separately. However, it is not trivial to robustly connect these approach so that the errors/failures of one is covered by the other, and to reflect sensor inputs to the exploration as soon as possible. An object searching problem based on bipedal humanoids has been proposed as a way to integrate these functions [7][8][9]. However, in recent methods, the search area has been limited by human[7], the approximate location of the object was given in advance[8], or the environment was completely known in advance[9][10]. Because these researches completely decompose the robot work into a planning phase, a sensing phase, and a walking phase, the robot is required to stop walking after arriving at each intermediary goal to get object/environment information. An automatic footstep planner on a point cloud has also been proposed in[11], but they did not consider autonomous exploration strategies.

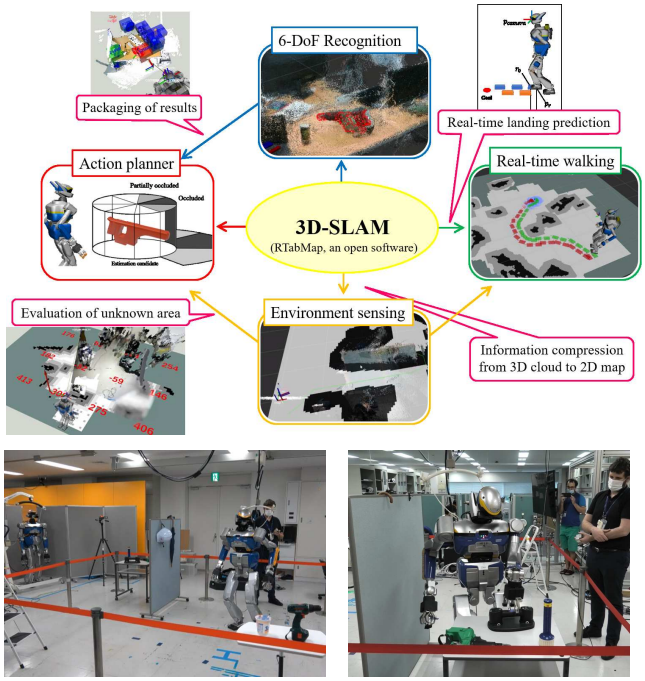


Fig. 1. Our humanoid robot HRP2-KAI achieved object grasping through an online exploration process and autonomous walking with 3D-SLAM.

In this paper, we propose a framework for an autonomous humanoid system integrating object recognition, environment

Manuscript received: October, 16, 2020; Revised: January, 12, 2021; Accepted: February, 1, 2021.

This paper was recommended for publication by Cesar Cadena Lerma upon evaluation of the Associate Editor and Reviewers' comments.

[1]CNRS-AIST JRL (Joint Robotics Laboratory), IRL http://jrl-umi3218.github.io/

[2]National Institute of Advanced Industrial Science and Technology (AIST), Japan

[3]Graduate School of Engineering Science, Osaka University, Japan

Digital Object Identifier (DOI): see top of this page.

recognition, action planning and bipedal walking, based on 3D-simultaneous localization and mapping (SLAM). In the proposed method, a head RGB-D camera is used to run the 3D-SLAM (RTabMap[12]), to map the environment in 3D, and to estimate its position in real-time. Simultaneously, our object recognition module detects a target object from the camera's input point cloud in a six degrees of freedom (6-DoF) format, and aggregates it with other information (such as the direction of observation) to memorize the recognition results in a world coordinate system, as a simple 3D probability map style. The 3D-SLAM module generates in real-time a 2D map that the robot uses for walking, by conservatively projecting obstacles from the point cloud onto a horizontal plane. Simultaneously, an "unknown" area in the 2D map (i.e., an area which cannot be observed due to obstacles) is used in our search strategy. An action planner module decides and updates the next goal in real-time based on a comprehensive judgment of multiple factors, such as the results of object recognition, the structure of the 2D map, the unknown areas which possibly contain the target object, and current position of the robot. A footstep planner module plans the walking path at a high speed. Our system predicts the next landing position of the foot based on a camera position estimated by 3D-SLAM, so that the robot can smoothly transition to the next walking plan without pausing when the map or goal is changed while walking.

Comparing to previous works, the novelty is that our exploration system can

- quickly reflect recognition results and automatically change the next goal even before the robot arrives at the previous goal.
- find a target object without providing hints in advance.
- reveal occluded/unobserved area from map structure.
- plan viewpoints based on accumulated results while walking.
- keep the goal planning time fast even after the map extends.

Therefore, our main contribution is a **stop-free** exploration framework: the robot can detect an object or obstacles, plan its next goal, and smoothly change its walking path to the new goal in parallel, without interrupting its walk. The main technical difficulty is that, as 3D-SLAM dynamically extends the map, it gets slower and slower to generate goal candidates for the whole map and to evaluate each utility one by one. To keep the planning speed fast, we implement a set of techniques, i.e., compact 3D probability map, cut off areas by heuristics, and inverse A* search.

In the following, we clarify the contributions and novelty of this research by introducing related studies (Section II), and explain how we established 3D-SLAM as the core of our system by providing the details of the methods (Section III). In Section IV, we describe several object search experiments using a real robot, and discuss the usefulness of our approach. We conclude with Section V.

II. RELATED RESEARCH

We suggest some related researches about object recognition, footstep planning, and autonomous robot related to exploration, object searching, or viewpoint planning.

A. Object Recognition

In this research, a 6-DoF format is desirable for object recognition, as the final goal is to grasp the target object. Research on object recognition is rapidly improving with developments in deep learning. The Yolo series[1][13] are not suitable for grasping tasks, as the output format is a 2D bounding box. The methods for recognizing objects in the 6-DoF format from RGB images [14][15][16][17] requires the correct answer data in the exact 6-DoF format for each training image. Point-Cloud recognition methods [2][18][19] have achieved object region extraction and categorization. However, as our research considered future construction work requiring accurate tool identification, we use 3D models to specify a detailed instance as a target object.

B. Footstep Planning

Recent studies [20][21][22] have performed reasonable planning with high reliability even in dynamic or 3D environments. R. Scona et al.[23] combines 3D-SLAM with humanoid robot walking. However, they decide a fixed footstep plan and correct online the walking motion to realize it. In contrast, we use the camera position estimated by 3D-SLAM for real-time estimation of feet landing positions, and always re-plan the footstep trajectory with the predicted landing point as the starting point, for every step. We partially reuse an open-source code from Humanoid Navigation[24] that we modified to handle the 2D obstacle map and start/goal position in real-time.

C. Exploration and View Planning

Saidi et al.[7][25] performed exploration with a humanoid robot. They use a heuristics to reduce the volume of non-observed space in a limited area. While they did not set any target objects, so it is sufficient that the robot has a brief look at every area, we set a specific target object and deal with an uncertainty of object recognition. B.Yamauchi[26] detects frontier areas in 2D map and the robot actively extends the environment map by going to the nearest one. In our case, we subsample the map in sub-divisions and evaluates them by a heuristics which consists of the size of unknown/obstacle area, accumulated recognition results, and a walking cost. Isler et al.[27] decide the next best viewpoint by the best information gain in a probabilistic volumetric map. In our method, after the selection of interesting object candidate, we generate viewpoint candidates surrounding it and select one based on an heuristic considering observed directions. Makarenko et al.[28] combined the exploration strategy[26] with the evaluation for stable localization in 2D-SLAM. The 3D-SLAM[12] we use is an color feature based tracking between camera images, and our exploration strategy does not take the possibility of localization lost into account. Gonzalez-Banos and Latombe[29] generate goal candidates for exploration based on boundaries of the 2D map. In our case, we sample goal candidates in circle, centered on a highly scored sub-division, and evaluate each goal candidate with considering observable area from there, walking cost by

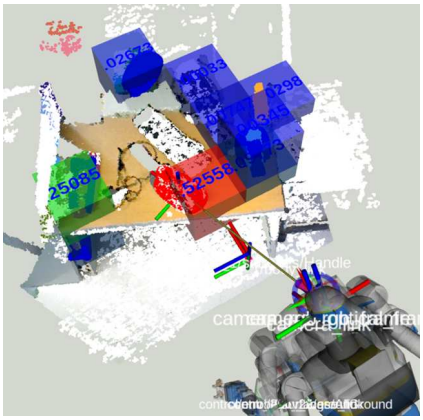


Fig. 2. Visualized result information containers in the world coordinate system. These boxes are located at each discretized key position. Each of them contains the ICP fitting score and observed directions, and is colored based on reference to ICP score.

reverse A* search. Masuzawa and Miura[30] express the time limit in the form of a penalty function. In comparison, our system continues to explore the area until the recognition module detects an object with an evaluation score high enough in our 3D probability map. Monica et al.[9] give a position of interest (POI) to the robot and their goal is to reduce occlusions around the POI area. In contrast, we do not provide any POI information nor hints, and the robot walks based only on its vision.

Kim and Likhachev[5] proposed an NBV planning method based on a POMDP. Their PR2 mobile robot detected a target object through a point cloud process from a cluttered and occluded table scene. In POMDP, the estimated object poses are described as hypotheses, that are validated or discarded by accumulating observations. Works in [31] and [32] are both using Markov assumption to estimate exploration utility for NBV planning to reduce occluded areas with mobile robots. However, their methods also require POIs or target regions in advance, to generate viewpoint candidates beforehand, and calculate utilities for every camera position candidate. They also require at least a few tens of seconds (e.g., 30s in [31]). In our case, the environment and object recognition results are updated in real-time; therefore, the exploration goal can change every time the robot obtains new information, even while walking.

III. OUR METHODS

A. Overview of System

In this paper, we present a scheme allowing a humanoid robot to autonomously find an object in an unknown environment. The inputs consist solely in the visual data acquired by a RGB-D camera and a 3d model of the target object. As shown in the upper part of Fig. 1, and further detailed in Fig. 3, we integrated four modules (6-DoF object recognition, environment recognition, action planner, and footstep planner) with 3D-SLAM. To connect the information obtained from the 3D-SLAM to the robot's autonomous activities, this research proposes four approaches for the integration of 3D-SLAM and each module.

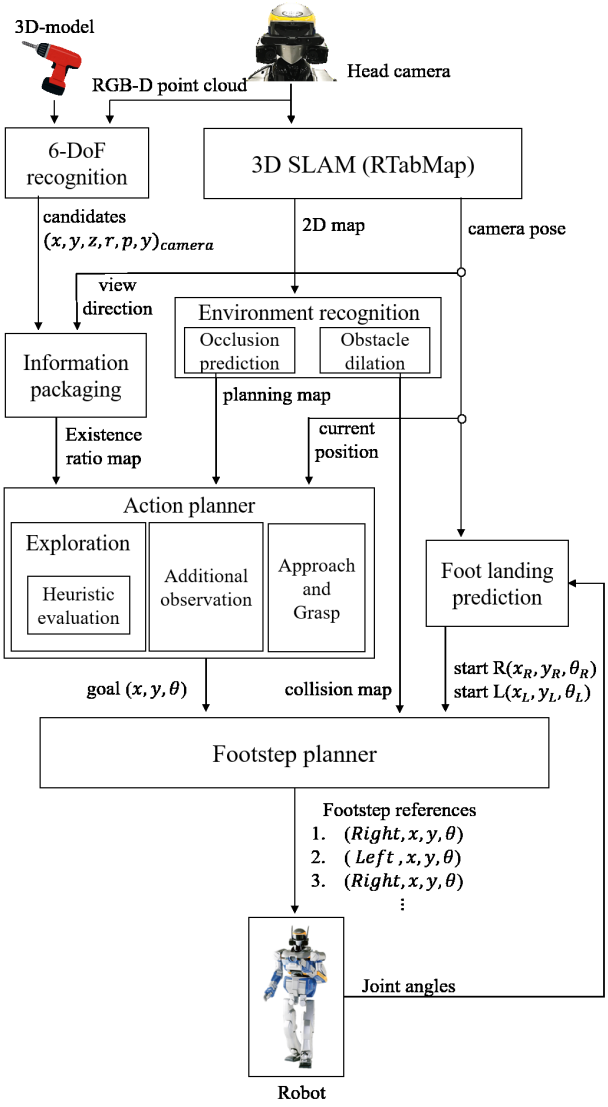


Fig. 3. Process flow of our system. 3D SLAM module obtains RGB-depth data through the Xtion camera and publishes an organized 3D point cloud, 2D map, and camera pose in a world frame. The modules all work in different threads.

The *6-DoF object recognition* module takes as inputs a 3D-model of the target object and an RGB-D point cloud from the camera, and estimates the object position in a 6-DoF format in real-time. The estimation results are aggregated and stored with additional information such as the observation direction and are embedded in the 3D probability map (Fig.2).

The *environment recognition* module divides the 2D map, which is a projection of the 3D point cloud map from the 3D-SLAM, into four areas: "obstacle", "collision", "floor", and "unknown". As this study uses the 2D map not only for walking but also for search action planning, the regions that are not observed yet because of obstacles are labeled as "unknown" and thus are preferentially observed to find the target object.

The *action planner* module constantly plans and updates the next destination in real-time, based on a comprehensive assessment of multiple factors such as the results of object

recognition, structure of the 2D environment map, and current position of the robot. As the search target may not always be in the robot's field of view from the beginning, we propose an action strategy based on heuristics for evaluating the priority of the search for each region. As the results of object recognition also include false positives, even if the robot is able to recognize an object once, it does not immediately perform a grasp. Rather, it determines the priority of observation by comparing it with other encapsulated recognition results, or plans additional observation actions from other directions.

The *footstep planner* module quickly plans the walking path to the target point suggested by the action planning module on the 2D map generated by the environment recognition module. Our system also predicts the landing position of the feet by using forward kinematics from the estimated camera position using the 3D-SLAM, and provides the planner module with the first step of a walking motion in order to shift smoothly to the next walking motion without pausing the walking motion when the goal is changed.

B. 6-DoF Object Recognition and Aggregation of Recognition Results

We construct the object recognition module based on feature-based matching. In addition, as the robot should ultimately grasp the object, the output is not in a bounding box format but rather a 6-DoF position $(x, y, z, roll, pitch, yaw)$, based on applying the 3D-model to the point cloud. The object recognition module divides the raw RGB-D point clouds obtained from the depth camera into many clusters based on color and distance, and for each cluster, it extracts Fast Point Feature Histogram (FPFH) features[33]. Then the module compares each feature to the target 3D-model's FPFH feature to estimate the 6-DoF position. Then, we apply iterative closest point (ICP) alignment, using the previous FPFH estimation result as the initial position. The module is always running in background, allowing the robot to detect the target object in real-time, even while walking.

However, the point cloud recognition module sometimes outputs false positives. Even worse, as the robot acquires points while moving, the object recognition results are not always exactly at the same position. To make the robot system robust, we uses a simple 3D probability map to accumulate the object recognition results (and the viewpoint planner module decides the next goal based on this probability map). Whenever the object recognition module outputs a result, the 3D probability map aggregates the ICP error and the camera direction into one result container that is stored. To be tolerant to the position error, we discretize the object's position into 30-cm cubic units, and if an object has been recognized in the vicinity in the past and an aggregation container already exists at the discretized position, we update the contents of the existing aggregation container instead of creating a new one. When updating, a higher ICP score is adopted, and the observation directions are added and expanded so that the objects are stored as having been observed from multiple directions. This overwriting process keeps the number of aggregation containers compact even after the robot walked around and the exploration area gets very large. Memorizing observed direction makes it possible for the robot to walk around the container position where the object has been detected multiple times with high accuracy and to make additional observations in directions not yet observed.

C. Information Compression from 3D Point Cloud to 2D Map

For the sake of generality and practicality, we perform the object search without a priory knowledge on the surroundings, such as a static 2D room map. We only suppose a flat ground. For this purpose, we designed an environment recognition module that recognizes the surrounding environment and expands the search area in real-time as the robot walks. Using the robot's head camera and the open-source 3D-SLAM software RTabMap[12], the robot maps the surrounding environment in a large-scale 3D point cloud format. However, the number of point clouds grows to hundreds of thousands as the robot walks; as such, the robot cannot maintain its processing speed if it accesses the whole point cloud every time when it plans the next reasonable viewpoint. In this paper, we use an original 2D map instead of a 3D map to retain the information for object search. This enables the humanoid robot to avoid obstacles, and to plan reasonable observation viewpoints under uncertainty.

Based on height and plane detection, RTabMap classifies the point cloud obtained from 3D-SLAM into three categories: floor, obstacle, and ceiling, . It also projects these clouds onto a 2D map, and two types of *floors* and *obstacles* are labeled. To prevent the robot from hitting obstacles while walking, our environment recognition module generates a "collision" area by dilation of the "obstacle" area by half the width of the robot. The footstep planner only allows the feet to land on the "floor", so that the robot avoids obstacles in real-time.

Additionally, to improve the efficiency of the object search, we make the robot prioritize unknown/occluded areas that have not yet been observed. Since there are many occlusions in the real world, the RGB-D camera cannot obtain any point cloud in an occluded area at all. Thus, the 3D-SLAM point cloud contains a significant amount of vacant space. However, we believe that vacant space is the most meaningful area for an object searching task in an unknown environment, as there may be a new "floor" area for leading robots to a further area, or the target object may be hidden. Therefore, our environment recognition module assigns a fourth type of area to such vacant spaces in the projected 2D map: "unknown". The action planner module determines goals by considering the "unknown" areas, and the robot walks to reveal such occluded information. It also adds "unknown" areas on the outside edges of the 2D map, enabling the robot to actively expand its range of searching.

D. Predictive Evaluation of Unknown/Occluded Area and Action Planner Module

Our exploration strategy consists of four phases: *large exploration*, *additional observation from another direction*, *approaching*, and *grasping*. This viewpoint planner module also runs in parallel and always updates the goal while the

Algorithm 1 Large Exploration Algorithm

Input: 2D environment map map, robot position p_{robot}, object candidates in recognition memory bank $o_1, .., o_J$
Output: next goal $g_{next}(x, y, \theta)$

```
separate map into 2m × 2m sub-divisions r_1, ..., r_I.
for r_i ∈ r_1, ..., r_I do
    compute "unknown" volume v_i(r_i)
    compute L2 norm L_i(p_robot, r_i)
    sub-division heuristic score h_i = w_1 v_i − w_2 L_i
    for o_j ∈ o_1, ..., o_J do
        if candidate o_j is inside of r_i then
            add bonus: h_i += w_3 score(o_j)
        end if
    end for
end for
sort sub-divisions r_1, ..., r_I by heuristic score h_1, ..., h_I
initialize the highest visiting utility u_max = 0
while g_next is empty do
    extrude r_high which has the highest utility score
    generate goal candidates g_1, ..., g_K around r_high
    for g_k ∈ g_1, ..., g_K do
        predict observable volume V_k(g_k, map)
        plan obstacle avoidance path A*(p_robot, g_k, map)
        get path cost c_k (if no paths found: c_k = ∞)
        u_k = V_k − c_k
        if u_k > u_max then
            update goal g_next = g
            update highest utility u_max = u
        end if
    end for
    remove r_high from the sub-division list r_1, ..., r_I
end while
return g_next
```

robot is walking, to immediately reflect the observation results in the exploration.

Large exploration

The exploration follows Algorithm 1. Because the robot does not have a complete nor perfect environment map, we have to plan the next goal heuristically. A unique feature of this research is that the heuristics evaluation equation includes not only the results of object recognition, but also the size of the "unknown" area as calculated by the environment recognition module. This allows the robot to operate with a policy of expanding its recognition area. It actively discovers hidden objects and new paths blocked by obstacles and performs exploratory actions that extend its own possible range of action. The biggest difficulty is to keep the planning time fast even after the area extends much larger, like $10m \times 10m$. The previous method[29] evaluates the observation utilities and reachability for every goal candidates. However, it gets much slower after the exploration area got larger. To keep the viewpoint planning fast, we first prioritize sub-divisions with heuristic scores, then generate goal candidates locally and dynamically as needed, and evaluate their utility and reachability (*l12* and *l16* in Alg.1).

The planner first divides the 2D map into $2m \times 2m$ sub-divisions (*l1* in Alg.1). For each sub-division, the area of the "unknown" and distance cost (L2 norm) from the current position are calculated and subtracted with weights to obtain a heuristics score (*l3* - *l5*). Furthermore, if there are object recognition result containers in the region, a bonus score proportional to its ICP accuracy is added to the heuristics score (*l6* - *l10*). These heuristics allow the robot to prioritize visits of area that are "unobserved", "close to the current position", and "where a likely object candidate has been detected in the past".

Once the sub-division of interest is determined, the next step is to determine the appropriate standing position (x, y, θ) to observe it. We generate discrete candidate goal points (on a 40-cm resolution grid, discarding all points in collision) inside and outside of the sub-division, and predict the observable area for each candidate point by considering the obstacle region in the 2D map. In addition, by searching the walking path from the current position to each candidate point on the 2D map by A* path search, the planner determines the reachability and calculates the walking cost. (If no path is found, the goal candidate point is rejected as unreachable.) To accelerate the planning time, we use reverse A* search and keep its closed list as a cache data indicating unreachable places. Because many candidate points are in small unreachable areas, this allows to quickly discard them (Fig.4). The above two points allow the robot to move around to a point that is unobstructed by obstacles, based on combining the area predicted to be observable with the distance cost of travel.

This exploration planning also runs in real-time (approximately 1 – 5Hz), so that the robot can update its search action by reflecting the results from object recognition and environment recognition while walking. When the recognition module finds an object with an ICP score higher than a threshold, the action planner module proceeds to the next "additional observations from another direction" phase to further verify the recognition results.

Additional observation from another direction

After the robot finds a score estimation result that is higher than a threshold, the robot attempts to confirm it. Owing to 3D-SLAM, our estimation result contains occluded/revealed direction information, so the robot is able to estimate and walk in the unobserved direction. Therefore, this observation strategy leads the robot to walk around the object candidate, and to observe it from several directions.

The planner first generates a number of goal candidates (x, y, θ) in a circle with a $2m$ radius area, centered on the recognition result on the 2D map. For each goal candidate, the planner checks whether any obstacles might disturb the observation by drawing a line from the goal to the estimated object position, and determines the next goal based on the weighted sum of the walking distance cost and observable direction which is scored as a decimal number between 0.0 and 1.0 (0.0 : perfectly observed, 1.0 : not observed at all). The black and white directions represent the observation score in Fig.5. After walking, if the camera module recognizes the

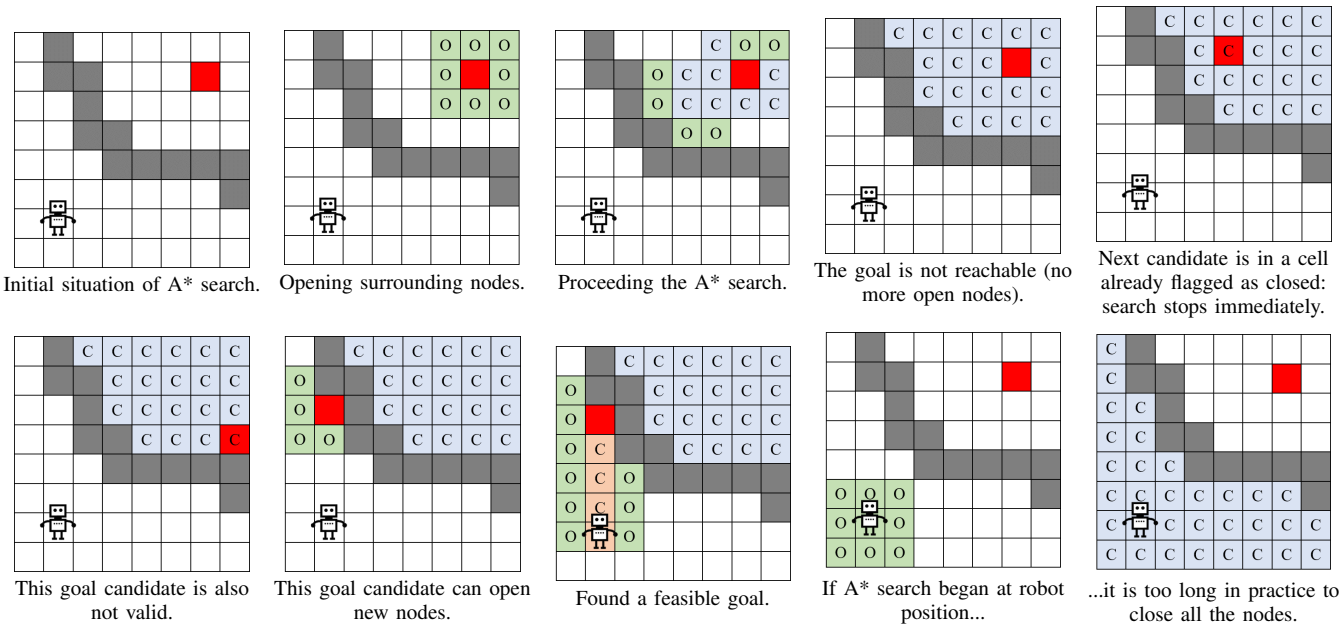


Fig. 4. Closed list as a cache memory in our A* search. Red : Goal candidate for which we want to check reachability, grey: obstacles, O : Open nodes, C : Closed nodes. Our system begins A* search from the goal candidate position, and re-uses the closed list for other goal candidates.

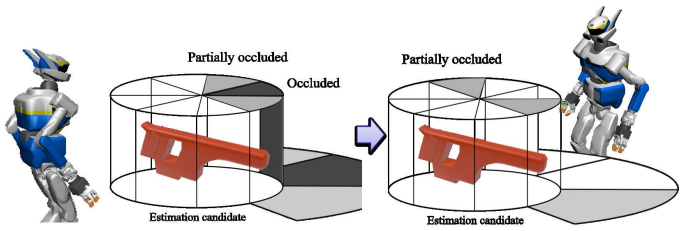


Fig. 5. Additional observation from another direction to confirm the recognition result.

object in the same position with a high score from more than two view directions, the system proceeds to the next strategy step, i.e., approaching. On the contrary, if the candidate does not achieve a good score from other viewpoints, the system reduces the probability of that result container, and resumes the large exploration step again.

Approaching Step

After confirming the object position from several view directions, the robot approaches the object as closely as possible to grasp the object. We use security margins for collision during the exploration, however, that could prevent the robot to come close enough for grasping. Therefore, we have to switch the collision check into more precise method. We design a pair of goals, named as "before approaching" and "final standing" at same time, and apply the precise collision check only to "final standing" goal.

First, the planner module designs a "final standing" position for grasping on the 2D map with only the object area. At this time, the planner uses precise collision checking which approximates the robot's 2D shape as a rectangle, instead of using collision area in the 2D map explained in Section III-C. The planner generates some feasible standing position candidates surrounding the estimated object position. Second, it straightly moves these "final standing" positions away by 50cm to get "before approaching" goal candidates. After confirming collision safety for each goals, the robot executes the pair of goals one-by-one. This allows the robot to approach object's support in a straight line, and to safely approach the edge of the object support.

Grasping After approaching the estimated object position, the robot uses a more accurate object registration method, i.e., 2D-6D local registration[34]. This local registration uses the 6-DoF estimated position from our recognition module as an initial guess and an iterative optimization process based on a black-and-white image to refine it. Upon completion, the robot selects the hand closer to the object and performs a grasping motion, following a spline-based trajectory of the hand defined with respect to the object position and a predefined grasp position manually chosen on the target 3D-model.

E. Real-Time Prediction of Foot Landing Position and Walking on Changing Map

To continue walking smoothly even if a goal is changed while walking, we propose a real-time landing position estimation based on 3D-SLAM (Fig.6). The 3D-SLAM module

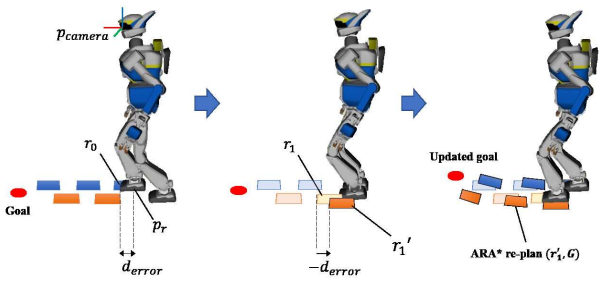


Fig. 6. Real-time prediction of foot landing position by 3D-SLAM.

always estimates a camera pose (p_c) in a world coordinate system, and a controller observes all joint angles of the robot. By solving whole-body forward kinematics, the robot can directly estimate its foot poses (p_r and p_l). To maintain stability, the robot does not change the next landing point while its foot is in air, and re-plans the footsteps plan from the next landing point. Assuming that the robot is standing on its right foot during walking, the previous landing point (p_r) is slipped by d_0 from the reference position r_0, and that error is added to the next left landing point r_1. The planner predicts the next landing point as $r'_1 = r_1 + d_0$; then, it starts re-planning from r'_1 on the left foot while the left foot is still in air. After the actual left foot landing, it calculates the actual landing point by solving the forward kinematics again, and obtains the error $d_1 = r'_1 - p_0$. This process enables the generation of a new footstep plan while the foot is in air (i.e., without waiting for foot landing), and a direct transition to a new walking motion. We did a brief experiment to verify its accuracy. (https://youtu.be/X74LgGnkgow)

IV. EXPERIMENTS

We applied the entire method on a real robot (HRP-2KAI) equipped with an XtionProLive RGB-D camera and attempted object-finding tasks in several environments. The implementation of the robot controller was done with *mc_rtc*[35]. The extrinsic camera calibration parameters were obtained using hand-eye calibration as proposed in [36]. We moved the robot's head link and obtained the relative motion of the camera from visual SLAM and the corresponding head link motion from the known robot's kinematics.

A. Scenario 1

We attempted a simple situation without obstacles. The target object was a vacuum gripper with size $30 \times 20 \times 11cm$. Poles were set up around the area to keep the robot inside the crane's range. The robot succeeded in grasping 8 times in 10 trials. (Evidence videos are available at https://youtu.be/al2jNbLTEL8 and https://youtu.be/CPV4_qupFic.)

B. Scenario 2

Second, we tried an occluded, more complex situation. We deployed a partition board in the area and hid the object on the other side of it. The robot could not see the object from the initial position at all, so it needed to walk to a meaningful viewpoint to reduce the occlusion.

The robot first looked around, and could not obtain any estimation results; then, it found the unobserved area and approached it. After walking to the other side of the partition board, it detected the target object in its sight and succeeded in grasping in the end. (See Fig.1 and the main video https://youtu.be/2rr84jnez4k)

C. Scenario 3

We tried unlimited exploration with a mobile lifter. The size of walkable area was approximately $8m \times 16m$, and it is a soft hourglass type, i.e., the robot has to find the path to the other side of area. (see Fig. 7 and a video https://youtu.be/q-xpZdomM0k)

D. Discussion

The different experiments validate our approach in that it let the robot find, walk to and grasp the target object. The grasping proves that the overall system is precise enough. The system is not perfect though, and while it often succeed, we sometimes encountered failures. For example the SLAM can get lost when in front of a largely un-textured wall or perturbed by a moving obstacle, or the ICP can fail to recognize an object. The most common cause of failure was that ICP would fall into a local solution and produce a higher match score than the truly correct object. "Additional observation" phase usually leads the robot to other view directions and the existence probability decreased, however, if there are no feasible viewpoints due to obstacles, the robot passes "additional observation" phase and tries to grasp the false positive result. There were also delicate parameter tuning problems. Weights for the exploration heuristics (*l5* and *l8* in Alg.1) changes the robot nature: e.g., giving big w_1 makes it more actively explore but it frequently travels back and forward over long distances, or giving big w_3 makes it cautious and check for low-scored result containers, and the map does not expand smoothly. The higher the threshold to change the exploration phase from "additional observation" to "approaching step", the more conservative the robot becomes and it takes a longer time to reach grasping step. We discretized the object's position into 30-cm cubic units to accept Point-Cloud registration errors, but the errors are relative to the size of the target object, and the discretization needs to be changed according to this size. As a thumb rule, We take the size of the cube so that it can slightly bound the object.

V. CONCLUSION

We proposed a way to use a 3D-SLAM system so that a humanoid robot can find automatically a desired object in an unknown environment. We experimented it on HRP-2KAI in three situations where the robot, deciding its next viewpoint by itself, succeeded in finding the object, that was not initially in its field of view, or was occluded by an obstacle.

ACKNOWLEDGMENT

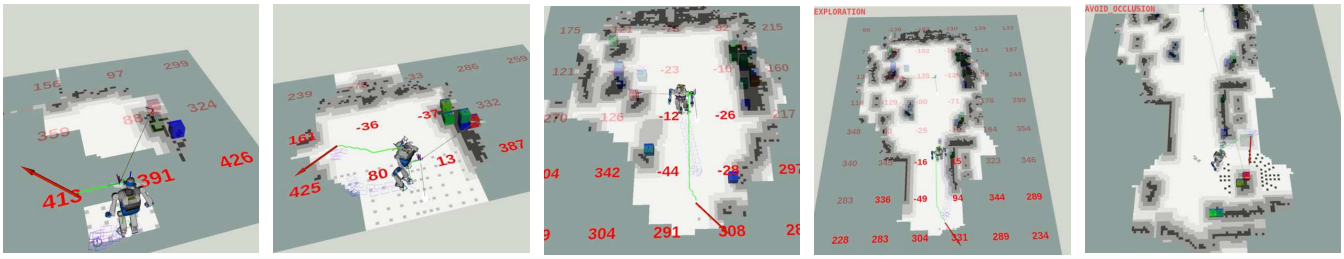
We thank Guillaume Caron for providing his 2D-6D local registration code[34], and Stéphane Caron who is the developer of the walking stabilization controller[37].

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conf. on Computer Vision and Pattern Recognition*, pages 779–788, June 2016.

[2] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2016. cite arxiv:1612.00593.

[3] N. Roy, P. Newman, and S. Srinivasa. *An Object-Based Approach to Map Human Hand Synergies onto Robotic Hands with Dissimilar Kinematics*, pages 504–. MIT Press, 2013.



0s : start creating map 5s : ignore false positives 55s : find new area 85s : extend known area 120s : find the object

Fig. 7. Example of the map building in the third scenario. The map is extended as needed. The score of every sub-division appears in red number. Division containing a large amount of "unknown" and being close to the current position results in high score, and leads the robot to observe there.

[4] X. Ye, Z. Lin, H. Li, S. Zheng, and Y. Yang. Active object perceiver: Recognition-guided policy learning for object searching on mobile robots. In *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 6857–6863, 2018.

[5] S.-K. Kim and M. Likhachev. Planning for grasp selection of partially occluded objects. In *IEEE Int. Conf. on Robotics and Automation*, 2016.

[6] J.Englsberger, C.Ott, and A.Albu-Schäffer. Three-dimensional bipedal walking control based on divergent component of motion. *IEEE Transactions on Robotics*, 31:355–368, 2015.

[7] F. Saidi, O. Stasse, K. Yokoi, and F. Kanehiro. Online object search with a humanoid robot. In *2007 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1677–1682, 2007.

[8] T. Foissotte, O. Stasse, A. Escande, and A. Kheddar. A next-best-view algorithm for autonomous 3d object modeling by a humanoid robot. In *8th IEEE-RAS Int. Conf. on Humanoid Robots*, pages 333–338, Dec 2008.

[9] R. Monica, J. Aleotti, and D. Piccinini. Humanoid robot next best view planning under occlusions using body movement primitives. In *2019 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2493–2500, 2019.

[10] S. Obwald, P. Karkowski, and M. Bennewitz. Efficient coverage of 3d environments with humanoid robots using inverse reachability maps. In *IEEE-RAS 17th Int. Conf. on Humanoid Robotics*, pages 151–157, 2017.

[11] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt. Footstep planning for autonomous walking over rough terrain. In *IEEE-RAS 19th Int. Conf. on Humanoid Robots (Humanoids)*, pages 9–16, 2019.

[12] M.Labbe and F.Michaud. Long-term online multi-session graph-based splam with memory management. *Autonomous Robots*, 42(6):1133–1150, 2018.

[13] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[14] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 4556–4565, 2019.

[15] Y. Xiang, T. Schmidtr, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *CoRR*, 06 2018.

[16] S. Zakharov, I. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. *Int. Conf. on Computer Vision*, pages 1941–1950, 2019.

[17] M. Rad and V. Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *2017 IEEE Int. Conf. on Computer Vision (ICCV)*, pages 3848–3856, 2017.

[18] C. Ruizhongtai Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30: Annual Conf. on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5099–5108, 2017.

[19] C. R. Qiharles, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *Proc. of the IEEE Int. Conf. on Computer Vision*, 2019.

[20] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz. Anytime search-based footstep planning with suboptimality bounds. In *12th IEEE-RAS Int. Conf. on Humanoid Robots*, pages 674–679, 2012.

[21] P. Michel, J. Chestnutt, J. Kuffner, and T. Kanade. Vision-guided humanoid footstep planning for dynamic environments. In *5th IEEE-RAS Int. Conf. on Humanoid Robots, 2005.*, pages 13–18, 2005.

[22] P. Karkowski, S. Oßwald, and M. Bennewitz. Real-time footstep planning in 3d environments. In *IEEE-RAS 16th Int. Conf. on Humanoid Robots*, pages 69–74, 2016.

[23] R. Scona, S. Nobili, Y. R. Petillot, and M. Fallon. Direct visual slam fusing proprioception for a humanoid robot. In *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1419–1426, 2017.

[24] J. Garimort, A. Hornung, and M. Bennewitz. Humanoid navigation with dynamic footstep plans. In *IEEE Int. Conf. on Robotics and Automation*, pages 3982–3987, 2011.

[25] F. Saidi, O. Stasse, and K. Yokoi. A visual attention framework for search behavior by a humanoid robot. In *6th IEEE-RAS Int. Conf. on Humanoid Robots*, pages 346–351, 2006.

[26] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE Int. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151, 1997.

[27] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza. An information gain formulation for active volumetric 3d reconstruction. In *2016 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3477–3484, 2016.

[28] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte. An experiment in integrated exploration. In *2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems(IROS)*, volume 1, pages 534–539 vol.1, 2002.

[29] G. B. Héctor and J. C. Latombe. Navigation strategies for exploring indoor environments. *Int. J. of Robotic Research*, 21:829–848, 10 2002.

[30] H. Masuzawa and J. Miura. Observation planning for environment information summarization with deadlines. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 30–36, 2010.

[31] C. Potthast and G. Sukhatme. A probabilistic framework for next best view estimation in a cluttered environment. *J. Visual Communication and Image Representation*, 25:148–164, 01 2014.

[32] A. Rasouli, P. Lanillos, G. Cheng, and J. Tsotsos. Attention-based active visual search for mobile robots. *Autonomous Robots*, pages 1–16, 08 2019.

[33] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE Int. Conf. on Robotics and Automation*, pages 3212–3217, 2009.

[34] N. Crombez, G. Caron, and EM.Mouaddib. 3d point cloud model colorization by dense registration of digital images. *ISPRS - Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5/W4:123–130, 02 2015.

[35] P.Gergondet. mc_rtc. In *code. https : //github.com/jrl − umi3218/mc_rtc*, 2015-2020.

[36] A. Tanguy, A. Kheddar, and A. I. Comport. Online eye-robot self-calibration. In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 68–73, 2018.

[37] S. Caron, A. Kheddar, and O. Tempier. Stair climbing stabilization of the hrp-4 humanoid robot using whole-body admittance control. *2019 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 277–283, 2019.