2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Visibility-based Viewpoint Planning for Guard Robot using Skeletonization and Geodesic Motion Model

Igi Ardiyanto and Jun Miura

Abstract— **This paper describes a viewpoint planning algorithm for a guard robot in an indoor environment. The viewpoint planner is used for the guard robot to watch a certain object such as human continuously. Rather than continuously follows the object, moving the guard robot using the viewpoint planner has many benefits such as reducing the movement and the energy used by the robot. Our viewpoint planner exploits the topology feature of the environment, which is extracted using a skeletonization technique to get a set of viewpoints. We search for escaping gaps from which the target may go out of the robot's sight, and make the movement model of the target and the robot to determine the predicted time of the worst case escape of the target. We then plan the action for the robot based on the geodesic model and escaping gaps. Simulation results using 3D simulator are provided to show the effectiveness and feasibility of our algorithm.**

I. Introduction

For supporting human life, the robot need to be close and interact with human. Generally, those closeness and interactions force the robot to have the ability for recognizing human and having the space awareness. For a specific need, the robot also has to be equipped with a specific ability, too. This ability is often taken or imitated from the human behavior when he faces the same task or situation. Let us take an example from a guardian who always watches a VIP officer in a gallery, or a cameraman who takes a film or documentation about a visiting official in a museum or offices. The guardian has to keep the VIP officer within his field of view without disturbing that person. Similar with the cameraman, he should always recognize the person without failing to keep that person within camera frames.

Our goal is to make a guard robot which imitates such jobs of the guardian or the cameraman. The robot is given a task to watch and capture the video of a person inside indoor environments like the museum, gallery, office, or exhibition room. Besides the main task of the robot which is to keep the person inside the video, the robot also should be aware of the space in the environment in order to take its advantages. For example, the robot can increase the efficiency of the batteries by taking the video while stopping at the point which has a large coverage. Another benefit is that the stopping robot can reduce noises and blur in the image frames due to the robot's instability when the robot moves.

We propose a viewpoint planning for a guard robot system in an indoor environments. Viewpoint is a point where the robot can safely watch the target by taking the video for a long time. The viewpoint planning gives a global plan for the robot to move while capturing the video of the target. The usage of viewpoints is to increase the possibility of the robot reducing its movement.

I. Ardiyanto and J. Miura are with Department of Computer Science and Engineering, Toyohashi University of Technology, Hibarigaoka 1-1, Tenpaku-cho, Toyohashi, Aichi, 441-8580, Japan {iardiyanto,jun}@aisl.cs.tut.ac.jp

II. Related Works

A. Art Gallery Problem

The art gallery problem is a problem of finding a minimal number of guards in a gallery with a complex polygonal shape so that every point inside the gallery can be seen continuously. This problem has been studied by many researchers, such as [13], [14], [15], and [16].

The art gallery problem is closely related to our problem, in terms of the ability of guarding an indoor environment. The main difference is that the art gallery problem uses several static guards to cover the whole area, while our guard robot problem has only one dynamic guard. We formulate our guard robot problem as the dynamic version of the art gallery problem, where the guard (i.e. the robot) will dynamically visit the possible static guards to maintain the coverage.

B. Pursuit-Evasion

Another problem that is close to our problem is the pursuit-evasion problem. In this problem, pursuer(s) and evader(s) move within the environment until the pursuer(s) locate(s) and catch all of evaders. There are also huge number of researches addressed to the pursuit-evasion problem, such as [8], [9], [10], [11], and [12]. In the classical pursuit-evasion problem, the target or the evader always tries to escape from the robot or the pursuer. Our approach assumes that the target moves independently while the robot always tries to locate the target. The uniqueness of our approach is that we use precomputed locations from which the robot can locate the target, and how the robot moves from one location to another using the smallest effort.

C. Object Following using Mobile Robot

Basically, the object following algorithm can also be used by a guard robot for tracking the target. The object following algorithm also has a long time history, such as the work by [17] and [18]. This technique will force the robot to always follow the target no matter the shape of the environment. Contrary, our approach tries to "understand" the shape of the environment and uses that information to track the target effectively. The viewpoint planner is also different from our previous work on people following robot [7], where the robot tries to be within some distances from the target, while the viewpoint planner tries to keep the target in the FOV.

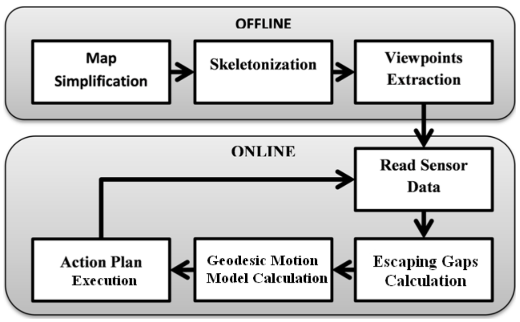


Fig. 1: Overview of viewpoint planning algorithm: action for extracting viewpoints in the offline stage (top), viewpoint planner execution in the online stage (bottom).

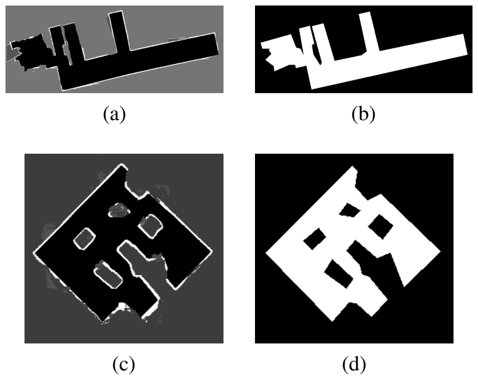


Fig. 2: The environment maps: the raw map from SLAM (left) and its simplified map (right).

D. Our Contribution

We aim to minimize the movement of the robot while always keeping the target within the camera frame (i.e. the robot field of view). Our contribution lies in the usage of viewpoints as the place for the robot for taking the video effectively and its movement planning. The viewpoint planner, which resembles the dynamic version of the art gallery problem, utilizes a new approaches using the skeletonization and geodesic motion model.

III. Skeleton-based Viewpoints

We divide the viewpoint planning algorithm into two stages: offline and online stages (see Fig. 1). In the offline stage, we examine the environment to get viewpoints. We then use those viewpoints to make action plans for the robot real-time, according to the current condition of the robot and the target.

To get viewpoints from the environment, we basically exploit the topology of the environment. We use the reasoning from human intuitions, for example in an indoor environment, the topology feature like the intersection is a place where we can stay for a long time to take the video because it covers a wide area (the intersection connects several corridors or hall way). Contrary, the corner of a room does not have such benefit like the intersection. We use the skeletonization of the map to get such topology.

The skeletonization technique itself is widely used in the image processing and pattern recognition applications (e.g. [20] and [21]). We introduce a new application of the skeletonization technique to get map topology and combine it with the template matching for extracting viewpoints from a map.

A. Environment Representation

Let us consider two dimensional grid map $\mathbb{C}$ which is obtained by a mapping algorithm (i.e. SLAM). This grid map consists of the passable area $\mathcal{F}$ and the non-passable area $\mathcal{N}$ for the robot. The grid map $\mathbb{C}$ which is retrieved by SLAM usually has a complicated shape (for example, see Fig. 2a and 2c). We simplify the map using three steps:

1) **Binarization**. Let $p_{x,y}$ be a point inside $\mathbb{C}$, the binary map $\mathcal{B}$ is simply obtained by using
$$\mathcal{B}(p_{x,y}) = \begin{cases} 1 & \text{for } p_{x,y} \in \mathcal{F} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$
2) **Contour extraction**. We extract the contour from the binary map using the method introduced by Suzuki, *et al.* [5], producing an outer boundary Γ and hole boundaries Γ_{H_i} with $i \in \{1, 2, \dots, n\}$, and both are closed polygonal chain. The number of holes n depends on the map (some maps may have no hole, $n = 0$, please see Fig. 2a for example).
3) **Polygon simplification**. Γ and Γ_{H_i} are then simplified by using Douglas-Peucker algorithm [6], to be a polygon Φ and holes H_i. Let $interior(\Phi)$ denote a set of $p_{x,y} \in \mathbb{C}$ which lie inside Φ, and $exterior(H_i)$ denote a set of $p_{x,y} \in \mathbb{C}$ which lie outside H_i. We then redefine the passable area $\mathcal{F} \subset \mathbb{C}$ in the environment as
$$\mathcal{F} = \{\forall p_{x,y} | p_{x,y} \in \{interior(\Phi) \cap exterior(H_i)\}\}. \tag{2}$$

We also define the robot model as a differential-steering robot given by $\mathbb{R} = \{x_r, y_r, \theta_r, v_r, w_r\}$, representing the robot position (x_r, y_r), heading θ_r, translational velocity v_r, and angular velocity w_r. An auto focus pan-tilt-zoom camera is attached on the robot, modeled by $\varphi = \{\varphi_{pan}, \varphi_{tilt}, \varphi_{zoom}\}$ representing pan, tilt, and zoom position respectively. Lastly, the target to be tracked is modeled by $\mathcal{O} = \{x_o, y_o, v_{xo}, v_{yo}\}$ where (x_o, y_o) and (v_{xo}, v_{yo}) are representing the target position and velocity.

B. Skeletonization

In an indoor environment, a person can easily recognize the environment's topology such as corridors, rooms, intersections, and corners. With such information, the person can determine which part of the environment can be used for fulfilling their needs; for example, the person will stay at an intersection to see connected corridors wider than at a corner. To imitate those human intuitions, we use map skeletonization to get such topological properties.

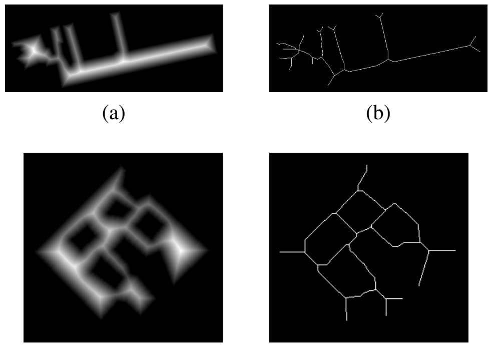

(a) (b) (c) (d)

Fig. 3: Map skeletonization: Distance Transform Map (left) and final skeleton map result (right).

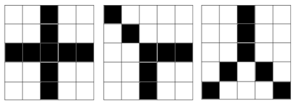


Fig. 4: Example of templates (5x5 cells).

Our skeleton map (see Fig. 3) is built using laplacian of distance transform. We first build a distance transform map $\mathcal{D}$ given by

$$\mathcal{D}(p_{x,y}) = \begin{cases} \sqrt{(x-x_2)^2 + (y-y_2)^2} & \text{for } (x,y) \in \mathcal{F} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where (x_2, y_2) is the nearest non-passable point to (x, y).

We then apply a laplacian filter to $\mathcal{D}$, to get the skeleton map $\mathcal{K}$, denoted by

$$\mathcal{K}(p_{x,y}) = \frac{\partial^2 \mathcal{D}}{\partial x^2} + \frac{\partial^2 \mathcal{D}}{\partial y^2}. \tag{4}$$

$\mathcal{K}$ is then binarized by a threshold (see Fig. 3b and 3d).

C. *Retrieving Viewpoints*

The skeleton map $\mathcal{K}$ gives us information about topology of the environment. We can see in Fig. 3b and 3d that $\mathcal{K}$ consists of endpoints, junctions, and connecting paths. We consider junctions as interesting points based on an assumption that we can gain wider field of view in such positions to watch the target. Endpoints are also considered as interesting points due to its ability to catch small details of the map like corners and the end of a corridor. We call those interesting points as **viewpoints**. We use template matching method over $\mathcal{K}$ map to get a set of viewpoints P. We use 30 templates simultaneously (see Fig. 4) and take the one with the smallest dissimilarity given by

$$d(x,y) = \sqrt{\sum_m \sum_n \left[\mathcal{W}(m,n) - \mathcal{K}(x+m, y+n)\right]^2} \tag{5}$$

where $d(x, y)$ is dissimilarity at point (x, y), $\mathcal{W}$ is the template map, (m, n) is size of $\mathcal{W}$, and $\mathcal{K}$ is the skeleton map. A viewpoint is said to be detected at location (x, y) (i.e. $p_{x,y} \in P$) when $d(x, y)$ is smaller than a predetermined threshold.

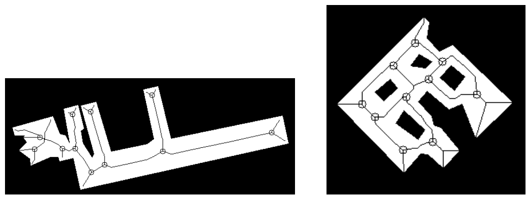

(a) (b)

Fig. 5: Maps with skeleton and viewpoints. Small circles denotes viewpoints.

D. *Visibility of Viewpoints and Its Optimization*

Visibility polygon is a polygon which encloses the visible area from a certain point. Let $V(P_i)$ denotes the visibility polygon of the point $P_i \in P$, $i \in \{1, 2, \ldots, k\}$ where k is the number of viewpoints. Visibility polygon $V(P_i)$ encloses the set of points in $\mathcal{F}$ which are visible from P_i, or we denote it as $interior(V(P_i))$. We use a similar idea with the art gallery problem that we want to minimize the number of viewpoints k such that the union of interior of visibility polygons will cover all of the passable area $\mathcal{F}$. This minimization problem is denoted by

$$f(k) = \bigcup_{i=0}^{k} interior(V(P_i)) \tag{6}$$

$$\arg\min_k f(k) \cong \mathcal{F}. \tag{7}$$

To solve (7), we do iterative pruning of viewpoints using two steps: (1) test each endpoint-type viewpoints (viewpoints which lie at the end of segments in the skeleton map), and prune it when $f(k) \cong \mathcal{F}$ is still satisfied by excluding that viewpoint, (2) do the same procedure for each junction-type viewpoints (viewpoints which lie at the intersection of segments). This order is based on an assumption that junctions usually have more field of view than endpoints. At the end of iterations, we redefine k as the optimized number of viewpoints (see Fig. 5).

In the case where the robot's visibility is limited (e.g. φ_{zoom} of the camera is limited), it may happen that $f(k) < \mathcal{F}$. We solve this problem by repeating the skeletonization and viewpoints retrieving on the area which is not covered before. By using this approach, we can make sure that $f(k) \cong \mathcal{F}$ is always satisfied.

IV. Viewpoint Planning

This section describes the viewpoint planning algorithm. The main goal of the viewpoint planning algorithm is to minimize the movement of the robot without losing the target. Here we assume that the camera tracking is done separately by another system (e.g., pan-tilt-zoom camera automatically adjusts the view toward the target), so that we can focus to the planning algorithm for the robot. As a consequence, we also assume that the robot has 360° field of view (FOV).

The viewpoint planning works as follows; First we search for escaping gaps from which the target may go out of the robot's sight. We then make the movement model of the

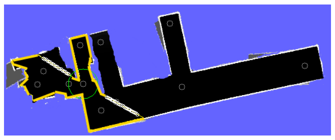


Fig. 6: Escaping Gaps. The small circle inside the green circle is the robot position and other small circles are viewpoints. The bold white line in the middle of free space (black area) is escaping gaps. Yellow lines denotes the visibility polygon.

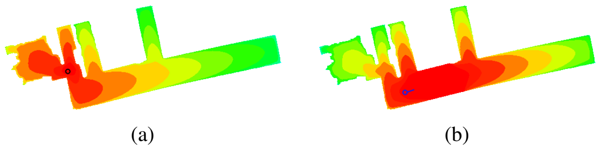


Fig. 7: The travel time map of the robot (a) and the target (b). The black circle represents the robot's current position. The blue circle with line denotes the target position and its predicted movement.

target and the robot to determine the the predicted time of the worst. Lastly, we plan the action for the robot based on the model and escaping gaps.

A. *Escaping Gaps*

Escaping gaps are a set of points from which the target may go out of the robot's sight. Escaping gaps have a similar idea with the well-known term *frontiers* for exploration of unknown space (e.g. [9]). With the same definition of $V(P_i)$, let $V((x_r, y_r))$ denotes the visibility polygon created by the current robot position (x_r, y_r). If $b(V)$ denotes a set of points which lie at the boundary of polygon V, then we define escaping gaps Λ as a set of points $p_{x,y} \in b(V((x_r, y_r)))$ which do not lie at the environment boundaries Φ nor H_i (see subsection III-A for the definition), or we can write it as

$$\Lambda = \{p_{x,y} | p_{x,y} \in b(V((x_r, y_r)) \wedge \neg(p_{x,y} \in b(\Phi) \vee p_{x,y} \in b(H_i))\} \tag{8}$$

Equation (8) can easily be understood by seeing Fig. 6. The bold white line in Fig. 6 represents escaping gaps which lie on the visibility polygon but do not lie on the environment boundaries.

B. *Geodesic Motion Model for Target and Robot Movement*

We use a worst-case assumption to ensure the target will not escape from the robot's view. The worst-case assumption is formulated based on two propositions[1]: (1) from the target point of view, the target is independent from the robot, i.e. the target does not care about the robot action, (2) from the robot point of view, the robot always thinks that the target will try to escape from the robot view. In other word, the robot thinks that the target tries to escape through the nearest escaping gap.

Based on the worst-case assumption, we compute the predicted time for both the robot and the target to reach all of escaping gaps. Computing the travel time by using Euclidean Distance (ED) is not a good option, because ED does not consider the shape of the environment. We prefer to use geodesic model by using the wave front approach of [1].

Let a monotonic wave front originated from a determined source point moves across a space, then the travel time T of the wave front in every point $p_{x,y}$ can be calculated using

$$|\nabla T(p_{x,y})| = \frac{1}{J(p_{x,y})}. \tag{9}$$

The travel time of a point depends on the distance from the source point and the velocity function $J(p_{x,y})$ used for traveling the wave front toward that point. This problem is known as *Eikonal equation* problem, and according to [1] eq. 9 can be approximated by first order finite difference scheme

$$\max\left(\frac{T(p_{x,y}) - T_1}{\Delta x}, 0\right)^2 + \max\left(\frac{T(p_{x,y}) - T_2}{\Delta y}, 0\right)^2 = \frac{1}{J(p_{x,y})^2} \tag{10}$$

where

$$\begin{aligned} T_1 &= \min\left(T(p_{x+1,y}), T(p_{x-1,y})\right) \\ T_2 &= \min\left(T(p_{x,y+1}), T(p_{x,y-1})\right) \end{aligned} \tag{11}$$

The solution[2] of (10) is given by

$$T(p_{x,y}) = \begin{cases} T_1 + \frac{1}{J(p_{x,y})} & \text{for } T_2 \geq T \geq T_1 \\ T_2 + \frac{1}{J(p_{x,y})} & \text{for } T_1 \geq T \geq T_2 \\ \text{quadratic solution of (10)} & \text{for } T \geq \max(T_1, T_2) \end{cases} \tag{12}$$

The velocity model of both the robot and the target is defined as follows:

1) For the robot, we want the robot to move safely in the environment. We use the distance map $\mathcal{D}$ in (3), then it is normalized to 0 and maximum speed of the robot v_{rmax} to give the velocity function

$$J_{robot}(p_{x,y}) = \|\mathcal{D}(p_{x,y})\|_{norm(0, v_{rmax})}. \tag{13}$$

It means we give a higher velocity in the area which is farther from obstacles (see Fig. 7a).

2) For the target, we take the current target velocity into account by using *velocity cone* model combined by distance map $\mathcal{D}$. Let $\mathcal{C}$ be the cone area which consists of points $p_{x,y}$ satisfying

$$\mathcal{C} = \left\{\forall p_{x,y} | p_{x,y} \in \mathcal{F} \wedge \left(\angle p_{x,y} \leq \arctan\left(\frac{v_{xo}}{v_{yo}}\right) \pm \frac{\pi}{3}\right)\right\}, \tag{14}$$

then velocity function J_C of the cone area is given by

$$J_C(p_{x,y}) = \begin{cases} v_{target} & \text{for } p_{x,y} \in \mathcal{C} \\ \varepsilon & \text{for } p_{x,y} \notin \mathcal{C} \wedge p_{x,y} \in \mathcal{F} \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

[1]Those propositions give us a slightly different definition of our algorithm compared to the pursuit-evasion problems. In the pure pursuit-evasion problem, the target or evader always tries to escape from the robot or pursuer rather than make an independent action.

[2]The quadratic solution in this equation means a quadratic equation $ax^2 + bx + c = 0$ has the solution $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

where $v_{target} = (v_{xo}^2 + v_{yo}^2)^{\frac{1}{2}}$ and ε is a small constant. We then combine J_C and $\mathcal{D}$ to get velocity model for the target

$$J_{target}(p_{x,y}) = J_C(p_{x,y}) \left\| \mathcal{D}(p_{x,y}) \right\|_{norm(0,1)}. \quad (16)$$

This combination is the key of our proposed geodesic motion model for the viewpoint planner. Figure 7b shows that by using velocity model, the travel time of the target will follow the shape of environment.

Each (13) and (16) respectively substitutes $J\left(p_{x,y}\right)$ in (10), then we get geodesic model of the travel time for the robot $T_{robot}(p_{x,y})$ and for the target $T_{target}(p_{x,y})$.

Back to the worst-case assumption, we try to find the most critical escaping gap $\lambda_{critical}$ (i.e. the fastest one which can be reached by the target) by examining

$$\lambda_{critical} = \arg\min_i T_{target}(\lambda_i) \quad (17)$$

for $i = (1, 2 \ldots, n)$, n is the number of escaping gaps, and $\lambda_i \in \Lambda$.

Basically, (17) also tells us that if the robot just stops at its position, it will lose the target at the predicted time $T_{target}(\lambda_{critical})$. It gives us an important definition,

Definition 1: The stopping robot will lose the target at $T_{target}(\lambda_{critical})$, except at a condition $T_{target}(\lambda_{critical}) \to \infty$.

It is imaginable that if the target moves away from all of escaping gaps (i.e. the target is always in the robot FOV), then based on the velocity model of the target, escaping gaps will have very small velocity which leads to a very large T_{target} (see eq. (10) and (15)).

C. Planning using Cost Minimization

To get a minimum amount of the robot's movement while keeping the target inside the robot's FOV, we use cost minimization for the planning. We define several properties for the planning algorithm:

- P as set of viewpoints,
- $\mathcal{S}$ as set of states,
- $\mathcal{A}$ as set of actions,
- $\mathcal{R}(\mathcal{S}, \mathcal{A})$ as applied rules based on $\mathcal{S}$ and $\mathcal{A}$,
- $\beta(\mathcal{S}, \mathcal{A}, \mathcal{R})$ as the cost caused by action $\mathcal{A}$, current state $\mathcal{S}$, and rules $\mathcal{R}$.

We use two states, $\mathcal{S} = \{s_0, s_1\}$, where s_1 is the state where the current position of the robot is at the one of possible viewpoints P and s_0 is the state where the robot is not at the viewpoint (i.e., the robot is moving from one viewpoint to another). We also define possible actions for the robot as $\mathcal{A} = \{a_0, a_{P_1}, \ldots, a_{P_k}\}$, where k is number of viewpoints, a_0 represents "do nothing" action for the robot[3] (i.e. the robot just stops at the current position), and $\{a_{P_1}, \ldots, a_{P_k}\}$ are the action for the robot to go to one of viewpoint P_i.

[3]The stopping action only describes that the robot stays at the same position, but actually the robot can make rotation or controlling the pan-tilt-zoom system to adjust its view toward the target, and it is also applicable for other actions.

We introduce the following rules for $\mathcal{R}(\mathcal{S}, \mathcal{A})$:

- **Rule 1**. The action a_0 is only applicable to the state s_1. This rule arises due to the definition of the viewpoint planning itself, where we want the robot to make a transition between viewpoints (i.e. the robot should not stop at non-viewpoint). It will cause a penalty cr_1, given by

$$cr_1(s, a) = \begin{cases} \infty & \text{for } s = s_0 \wedge a = a_0 \\ 0 & \text{otherwise.} \end{cases} \quad (18)$$

- **Rule 2**. An action which leads to the viewpoint where the robot cannot see the critical escaping gaps is not applicable. It is understandable that such viewpoints will make our robot lose the target. This rule causes a penalty cr_2, given by

$$cr_2(a) = \begin{cases} 0 & \text{for } a \in \\ & \{a_i | \lambda_{critical} \in interior(V(P_i))\} \\ \infty & \text{otherwise.} \end{cases} \quad (19)$$

  Please see the definition of $interior(V(P_k))$ at subsection III-D.
- **Rule 3**. This rule is a consequence of "Definition 1". Let $P_{critical}$ be a viewpoint inside the visibility polygon of the robot $V((x_r, y_r))$ (see the definition in subsection IV-A) which is the nearest to the critical escaping gap. The rule is based on a simple intuition that if the time for the robot to reach the viewpoint $P_{critical}$ is longer than the time for the target to reach the critical escaping gap $\lambda_{critical}$, then the robot will lose the target, or we can write it as

$$T_{robot}(P_{critical}) \leq T_{target}(\lambda_{critical}) \quad (20)$$

  Equation (20) happens on the stopping action, but it is also applicable for other actions. This rule causes a penalty cr_3, given by

$$cr_3(a) = \begin{cases} 0 & \text{for all } a \text{ when} \\ & T_{robot}(P_{critical}) \leq 0.8\mathbb{T} \\ 0 & \text{for } a = a_{P_{critical}} \text{ when} \\ & 0.8\mathbb{T} \leq T_{robot}(P_{critical}) \leq \mathbb{T} \\ \infty & \text{otherwise,} \end{cases} \quad (21)$$

  where $\mathbb{T} = T_{target}(\lambda_{critical})$. Basically, eq. (21) elaborates eq. (20) to see if the robot is at the critical time for losing the target. Before this critical time condition is violated, the robot can choose any action including the stopping action. When the critical time for losing the target is near, the robot has to do an action for preventing it (i.e., the robot should go to $P_{critical}$), because if the robot does not do anything then the condition will be violated, and any action cannot help the robot from losing the target (see the third row of eq. (21)). A constant 0.8 (experimentally obtained) is given to make sure the robot does not violate the critical time (i.e. ensuring the robot to move before violating the critical time).

Putting them together, we define the sum of the penalties caused by the rules as

$$\mathcal{R}(s,a) = cr_1(s,a) + cr_2(a) + cr_3(a), \text{ for } a \in \mathcal{A}, s \in \mathcal{S} \tag{22}$$

We then exclude all actions which give $\mathcal{R}(s,a) \neq 0$ from $\mathcal{A}$.

Finally, we select the action based on the minimization problem of the cost $\beta(\mathcal{S},\mathcal{A},\mathcal{R})$, given by

$$\arg\min_a \beta(s,a,\mathcal{R}), \text{ for } a \in \mathcal{A}, s \in \mathcal{S} \tag{23}$$

where

$$\beta(s,a,\mathcal{R}) = T_{robot}(P_i), \tag{24}$$

P_i is the viewpoint selected by action a_{P_i}. To summarize, the behavior of the guard robot will be as follows:

- When the robot is at a viewpoint, it will stay there within the viewpoint until the condition in rule 3 is violated.
- When the robot is not at a viewpoint (i.e. the robot is moving from one viewpoint to another), the robot will not stop until it reaches one viewpoint.

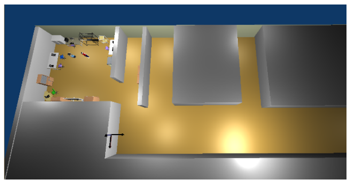
After an action was selected, the path which leads to the goal of the action is extracted as the planning result. The path is calculated by backtracking the geodesic motion model of the travel time for the robot $T_{robot}(p_{x,y})$ from the goal (chosen viewpoint P_i) to the robot position $p(x_r, y_r)$. We then send the planning result as a set of waypoints to a local path planner to be executed. We use path planner algorithm in [7] as the local path planner.

V. Experiments and Discussions

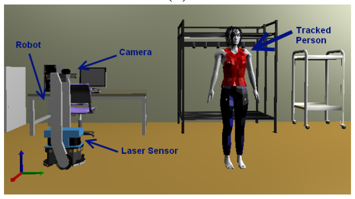
We test the viewpoint planning algorithm in a 3D simulation representing the real robot and environment. We implement our viewpoint planner as an RT-Component which is software module running on RT-Middleware [19] environment. We use a 3D simulator ([3], [4]) to perform the guard robot system consisting of the viewpoint planner and the person tracking using a pan-tilt-zoom camera.

We use a color-based particle filter for tracking a person in a red clothing. This tracking system uses a simulated pan-tilt-zoom camera and runs independently from the viewpoint planner; it behaves like the camera for cameraman or the eyes of the guardian which continuously captures and keeps the tracked person within the camera frame or the guardian FOV. The viewpoint planner runs in two stages: offline and online stages. In the offline stage, we get the map data from a SLAM algorithm, then we retrieve viewpoints from the map using the skeleton-based algorithm. We then use these viewpoints to make a global plan for the robot in the online stage. The action chosen from the global plan are then executed by a local planner.

The 3D simulator represents the first floor of ICT building of our university (see Fig. 8a). We use the simulated robot, a laser range finder, a camera, and environments mimicking

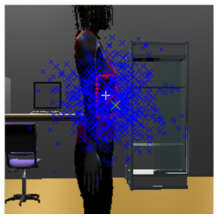


(a)

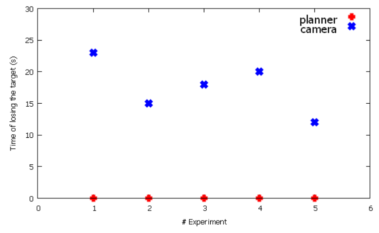


(b)

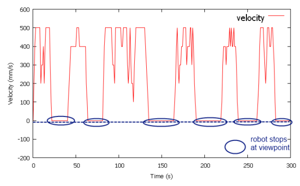
Fig. 8: 3D simulator appearance: (a) top view of the environment, (b) robot and target appearances.



(a)



(b)



(c)

Fig. 9: Result of the guard robot simulations: (a) Tracked person, (b) Time of losing the target, (c) Velocity profile of the robot, blue ellipses indicate the state when the robot stops at a viewpoint.

the real condition (see Fig. 8b). To show the robustness of the algorithm, we perform the guard robot simulation five times, and capture the time of losing the target according to the viewpoint planner and the camera frame. Better planner will have less time for losing the target. We also record the velocity of the robot during simulation.

Figure 9b explains the time of losing the target. As we can see that according to the planner, our algorithm never loses the target, but the camera sometimes loses the target. This behavior happens because of a late response of the pan-tilt system inside the 3D simulator. Figure 9c shows the velocity profile of the robot. The zero velocity means the robot stays in one place to track the person, while others mean the robot moves from one viewpoint to another viewpoint. In this case, the more the robot stays in one place, the less energy used by the robot to move, which means we will get more energy saving as the result.

We then test the viewpoint planning algorithm in a more challenging environment which has many rooms (in the geometrical terms, it is called polygon with holes, please see Fig. 10). Our algorithm can produce feasible viewpoints

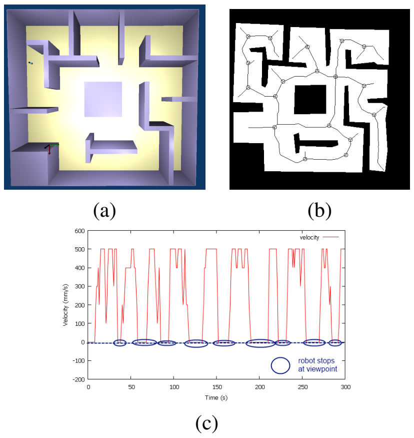


Fig. 10: Simulation in more complex environment: (a) top view of the environment, (b) Skeletonization and viewpoints of the map, (c) Velocity profile of the robot, blue ellipses indicate the state when the robot stops at a viewpoint.

for the robot (see Fig. 10b) for not losing the target. Figure 10c also tells us that the robot stays at the viewpoints which leads to energy saving.

We compare the performance of our viewpoint planner with the ordinary people tracking algorithm as described in [7], to see the effectiveness of the robot's movement. Table I shows the energy used by the robot for each algorithm. Simulation 1 refers to the simulation in Fig. 8, while simulation 2 refers to the simulation in Fig. 10. The energy is calculated by the integration of the translational and rotational energy, given by

$$Energy = \sum_{k=0}^{T} \frac{1}{2} m v_k^2 + \sum_{k=0}^{T} \frac{1}{2} I \omega_k^2 \qquad (25)$$

where T is the total time, m is the robot mass, I is the moment of inertia of the robot, v is the translational velocity, and ω is the rotational velocity. We can see that our viewpoint planner uses less energy, which means it can reduce the movement of the robot, comparing to the ordinary people tracking algorithm.

TABLE I: Energy Comparison (in Joule)

	Viewpoint Planner	People Tracking
Simulation 1	561.75	724.50
Simulation 2	882.50	1056.05

VI. Conclusion

We have presented a novel visibility-based viewpoint planning algorithm for the guard robot using a skeletonization and a geodesic motion model. We utilize the topology of the environment to make an effective movement of the robot. Simulation results show that our algorithm can reduce the movement of the robot, thereby saving the energy and reducing the blur due to unstable attached camera.

Several improvements can be implemented; one of them is to examine the environment in the 3D space to determine viewpoints. It will give us a more robust planning for several partial occlusion cases, for example, when the target is partially occluded by a table in a dead-end corridor, the robot does not need to move to other viewpoints.

References

[1] M.S. Hassouna, A.E. Abdel-Hakim, and A.A. Farag. "PDE-based robust robotic navigation". Image and Vision Computing, vol. 27, pp. 10-18, 2009.
[2] J. Sethian. "A fast marching level set method for monotonically advancing fronts". In Natl. Academy of Sciences, vol. 93, pp. 1591-1595, 1996.
[3] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan. "Modular Open Robots Simulation Engine: MORSE". In Int. Conf. on Robotics and Automation, pp. 46-51, 2011.
[4] I. Ardiyanto and J. Miura. "RT Components for using MORSE Realistic Simulator for Robotics". The 13th SICE System Integration Division Annual Conference, pp. 535-538, 2012.
[5] S.Suzuki and K.Abe. "Topological structural analysis of digital binary image by border following". CVGIP, vol. 30, pp. 32-46, 1985.
[6] D.H. Douglas and T.K. Peucker. "Algorithms for the reduction of the number of points required to represent a line or its caricature". The Canadian Cartographer, vol. 10, pp.112-122, 1973.
[7] I. Ardiyanto and J. Miura. "Real-time navigation using randomized kinodynamic planning with arrival time field". Robotics and Autonomous Systems, vol. 60, no. 12, pp. 1579-1591, 2012.
[8] B.P. Gerkey, S. Thrun, and G.J. Gordon. "Visibility-based Pursuit-evasion with Limited Field of View". Int. Journal of Robotic Research. vol. 25, no. 4, pp. 299-315. 2006.
[9] J.W. Durham, A. Franchi, and F. Bullo. "Distributed pursuit-evasion with limited-visibility sensors via frontier-based exploration". In Int. Conf. on Robotics and Automation, pp. 3562-3568, 2010.
[10] L. Guilamo, B. Tovar, and S. LaValle. "Pursuit-evasion in an unknown environment using gap navigation trees". In Int. Conf. on Intelligent Robots and Systems, pp. 3456-3462, 2004.
[11] L. Guibas, D. Lin, J.C. Latombe, S. LaValle, and R. Motwani. "Visibility-based pursuit evasion in a polygonal environment". Int. Journal of Computational Geometry Applications, vol. 9, no. 5, pp. 471-494, 1999.
[12] V. Isler, S. Kannan, and S. Khanna. "Randomized pursuit-evasion in a polygonal environment". IEEE Trans. on Robotics, vol. 21, no. 5, pp. 875-884. 2005.
[13] L.H. Erickson and S.M. LaValle. "An Art Gallery Approach to Ensuring that Landmarks are Distinguishable". Robotics: Science and Systems, 2011.
[14] D. Avis and G.T. Toussaint. "An efficient algorithm for decomposing a polygon into star-shaped polygons". Pattern Recognition, vol. 13, no. 6, pp. 395-398, 1981.
[15] J. O'Rourke. "Art Gallery Theorems and Algorithms". Oxford University Press, 1987.
[16] S.K. Ghosh. "Approximation algorithms for art gallery problems", Proc. Canadian Information Processing Society Congress, pp. 429-434, 1987.
[17] D. Schulz, W. Burgard, D. Fox, and A.B. Cremers. "People tracking with mobile robots using sample-based joint probabilistic data association filters". Int. J. Robot Research, vol. 22, no. 2, pp. 99-116, 2003.
[18] N. Bellotto and H. Hu. "People Tracking with a Mobile Robot: a Comparison of Kalman and Particle Filters". 13th IASTED Int. Conf. on Robotics and Applications, 2007.
[19] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.K. Yoon. "RT-middleware: Distributed component middleware for RT (robot technology)". In Int. Conf. on Intelligent Robots and Systems, pp. 3555-3560, 2005.
[20] H. Fujiyoshi, A. J. Lipton, and T. Kanade. "Real-time human motion analysis by image skeletonization". IEICE Trans. Information Systems, vol E87-D, no. 1, pp. 113-120, 2004.
[21] J. Hsieh, Y. Hsu, H.M. Liao, and C. Chen. "Video-Based Human Movement Analysis and Its Application to Surveillance Systems". IEEE Trans. on Multimedia, vol. 10, no. 3, 2008.