ARL-TR-9657 • Mar 2023

DEVCOM ARMY RESEARCH LABORATORY

Context-Aware Visual Search Using a Pan-Tilt-Zoom Camera

by Philip David, André Harrison, Ramavarapu Sreenivas, and Joshua Whitman

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9657 ● MAR 2023

Context-Aware Visual Search Using a Pan-Tilt-Zoom Camera

by Philip David and André Harrison
DEVCOM Army Research Laboratory, Army Research Directorate

Ramavarapu Sreenivas and Joshua Whitman
University of Illinois at Urbana-Champaign

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
March 2023	Technical Report	July 1, 2020 - June 30, 2022

4. TITLE AND SUBTITLE
Context-Aware Visual Search Using a Pan-Tilt-Zoom Camera

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Philip David, André Harrison, Ramavarapu Sreenivas, and Joshua Whitman

5d. PROJECT NUMBER
W911NF-17-S-0003

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLA-JB
Adelphi, MD 20783

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-9657

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
primary author's email: <philip.j.david4.civ@army.mil>.

14. ABSTRACT
There is a need to perform visual surveillance in large indoor and outdoor environments where regions of interest (ROIs) may span horizontal views up to 360° and possess very large depths of field (e.g., 1 km or more). While in some scenarios an Internet-of-Things (IoT) approach allows multiple distributed cameras to cooperatively survey a large ROI, other scenarios, such as surveillance by a mobile robot, require the ROI to be surveyed by a small number of collocated cameras. In the latter scenarios, existing dual-camera methods that use a wide-angle camera to cue a pan-tilt-zoom (PTZ) camera are inadequate as the system is unable to observe very distant objects and events of interest in the wide-angle camera imagery, and therefore unable to cue the PTZ camera. To our knowledge, we are the first to address the surveillance problem of resource-constrained PTZ camera search for objects unobservable in wide field of view imagery. In this report, we first motivate and formally define the problem. We next propose solutions that use a single PTZ camera to intelligently and efficiently search a large ROI for objects of interest. Finally, we present results of experiments performed in simulated 3D environments.

15. SUBJECT TERMS
Military Information Sciences, PTZ camera, visual search, surveillance, small target detection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Philip J David
Unclassified	Unclassified	Unclassified	UU	57	19b. TELEPHONE NUMBER (Include area code) 301-394-5603

Contents

List of Figures

List of Tables

1. Introduction

Surveillance cameras are becoming more commonplace in public environments, as well as finding use in private security and military operations. We are particularly interested in scenarios where a single pan-tilt-zoom (PTZ) camera is used to perform surveillance in large outdoor environments, which may include 360° horizontal coverage and depths out to 1 km or more. These scenarios exist in many environments such as security for building exteriors, airports, highways, parking lots, and property perimeters; anomaly detection in dense urban environments; and surveillance in military overwatch missions. In environments with many vertical obscurations (e.g., trees and buildings), ground-based cameras will need to be carefully located to provide long-range views. As the elevation of the camera is increased above the ground level, by placement on tall poles or building rooftops, for example, obtaining views of distant regions becomes easier.

Imagery from these cameras may be used to detect objects and events of interest, either via human operators or automated processing. Many of these cameras have the capability for an operator to pan, tilt, and zoom the camera optics to view focused regions of an environment in high detail, allowing both wide-area and long-range surveillance. With the proliferation of surveillance cameras, however, it is becoming impracticable for human operators to monitor and control all installed cameras. Intelligent, automated algorithms are needed to replace the human operators. It is not sufficient to simply repeatedly scan an environment with a PTZ camera set to a high zoom, as this is unnecessary for parts of the environment in the near field as well as in areas that cannot possibly contain objects and events of interest. Furthermore, the time to perform a high-zoom scan is excessive and may cause long delays in revisit times, increasing the chances of missing important events.

We introduce here the formal PTZ Search Problem (PTZSP). In the PTZSP, a fixed PTZ camera starts at a given PTZ position and must move through the continuous 3D space of PTZ coordinates to detect as many objects of interest as possible as quickly as possible. Reward is given for true positives, deducted for false positives and false negatives, and deducted for the time spent performing the search. When the PTZ space is discretized, the PTZSP is closely related to the Orienteering Problem with Functional Profits (OPFP).[1] However, the huge size of the discretized

search space makes the problem intractable for all existing OPFP algorithms.

There are no existing benchmarks on which to develop, evaluate, and compare algorithms for the PTZSP. Creating such benchmarks is difficult because each run of an algorithm may acquire a different set of images of the environment, and one cannot expect to record in advance all possible views of a real environment. To remedy this problem, we have created a high-fidelity simulation of a state-of-the-art object detector[2] as it would perform on real color camera imagery, and have integrated this with a small and accessible simulation of PTZ cameras operating in large outdoor environments. The combination enables researchers an easy means to develop and compare algorithms for the PTZSP. We present details of the simulations, describe three algorithms for solving the discretized PTZSP, and then analyze the performance of each algorithm on a set of 100 reproducible test cases.

2. Related Work

Dual-Camera Systems

Dual-camera (a.k.a., master-slave) systems (e.g., Hu et al.,[3,4] Scotti et al.,[5] and Tarhan and Altug[6]) use two cameras, the first being a fixed, wide-angle camera that performs initial object detection over the full region of interest (ROI) and then cueing of a second camera, a PTZ, to obtain higher-resolution data on detected objects. The main drawback of dual-camera systems is that there may be too few pixels on distant objects for reliable detection in the fixed, wide-angle camera images using appearance-based methods (e.g., you only look once [YOLO]), so other features, such as motion, are necessary. But then, distant, stationary targets will likely not be cued for viewing by the PTZ camera.

Hierarchical Object Detection

Some existing object detection approaches search for objects in a given image in a hierarchical, coarse-to-fine fashion using a virtual PTZ camera. For example, Bueno et al.,[7] Wu et al.,[8] and Caicedo and Lazebnik[9] use deep reinforcement learning to train an agent to iteratively refine the position of an object bounding box. Wu et al.[10] detect airplanes and ships in high-resolution satellite images by first detecting airports and seaports in a reduced resolution image, and then focusing attention around these regions for the plane and ship detection. These problems differ from ours in that the image data is static throughout the search process and there is no consideration for the time required to change the focus of attention. More

importantly, they do not require simultaneous solution of shortest path problems, which are believed not to have any polynomial-time solutions.

Information-Theoretic Scene Exploration

Sommerlade and Reid[11] and Salvagnini et al.[12] use information-theoretic approaches for PTZ camera surveillance. These methods balance the conflicting demands of exploration versus exploitation by maximizing expected information gain on each camera observation, but they do not address time constraints due to physically changing PTZ camera parameters.

Orienteering with Functional Rewards

The classic orienteering problem (OP) is a routing problem where the goal is to find a route that visits a subset of available nodes such that the route collects maximum reward but does not exceed a specified length or time budget.[13–17] Mukhina et al.[1] extend this problem to one with *functional profits* (OPFP), where the score of a specific node depends on its position in the route and on what other nodes are also in the route. One may consider discretizing the 3D PTZ space of the PTZSP into a finite set of nodes and then applying OPFP algorithms. Unfortunately, the size of the discretized PTZ space may be 10 million nodes or more,* which would make application of existing OPFP algorithms infeasible.

Learning Where to Look in High-Resolution Images

A few previous works have addressed the problem of learning where, in a high-resolution image, to look and process in order to achieve computational efficiencies without sacrificing accuracy of classification or detection tasks. Some methods[18,19] apply supervised learning with human gaze data to learn focus of attention models. Others[20–22] learn similar models using reinforcement learning. All of these approaches assume that a complete high-resolution image is provided at the start and that there is no time penalty for moving the focus from one area to another, two assumptions which are not valid for PTZ search problem.

*To allow full flexibility in selection of PTZ camera positions, all discrete pan-tilt positions must be allowed independent of camera zoom. It is up to the planning system to select the most informative PTZ positions. Therefore, a $180^\circ \times 45^\circ$ horizontal by vertical field of regard, quantized into 0.1° cells, with 10 discrete camera zoom settings, would result in $1800 \times 450 \times 10 = 8,100,000$ nodes.

3. Approach

In this section, we first formalize the particular PTZ visual search problem that we address (Section 3.1), and then describe simulations that are used to train and evaluate proposed algorithms (Section 3.2). This is followed (in Sections 3.4–3.6) by descriptions and pseudo-code of four visual search algorithms: a quadtree search, two variations of a genetic algorithm search, and a traveling salesman problem (TSP) search. The latter three of these four algorithms use estimated depth to predict optimal camera zoom across ROIs. Our approach for predicting optimal zoom is described in Section 3.3.

3.1 Problem Formulation

The visual search system, using a single PTZ camera, attempts to maximize target detection precision and recall of all *PTZ-detectable* targets within a fixed range of PTZ camera pan $\psi \in [\psi_{\min}, \psi_{\max}]$, tilt $\theta \in [\theta_{\min}, \theta_{\max}]$, and zoom $z \in [z_{\min}, z_{\max}]$, while simultaneously minimizing the total time to acquire and process these images. We define a target to be *PTZ-detectable* if there is some valid camera pan, tilt, and zoom setting producing an image in which the target can be detected with high confidence. For the current work, we assume that the PTZ camera is stationary (other than pan, tilt and zoom), and that all targets in the environment are stationary. This latter assumption means we do not have to be concerned with tracking moving objects.

A PTZ camera's zoom is adjusted by changing the focal length of its lens, which results in changes to the camera's horizontal and vertical fields of view. The horizontal field of view (HFOV) of this camera is an angle $\zeta \in [\zeta_{\min}, \zeta_{\max}]$. The vertical field of view (VFOV) is determined given the camera's HFOV and its image aspect ratio. For the remainder of this report, instead of using "zoom" to reference camera focal length, we use it to mean the camera HFOV. The use of HFOV instead of focal length allows more concise specification of our algorithms.

A *local environment model*, $\mathbb{M}$, is a collection of cylindrical panoramic images, $\left\{\boldsymbol{I} \in \mathbb{R}^{\mathrm{P}\times\mathrm{T}}\right\}$, describing properties of the local environment $\mathbb{E}$ that are partially observable by the PTZ camera. The panoramics in $\mathbb{M}$ may be created by mosaicking imagery derived from the planar PTZ camera images.[23] These images are indexed by discrete pan and tilt coordinates, (p, t), and each image spans the entire region of

interest of the visual search. We assume that each cell $\boldsymbol{I}[p,t]$ of a model panoramic represents some property of a $\triangle s \times \triangle s$ solid angular region of the environment. $\mathbb{M}$ may include information such as terrain class, depth, target presence, as well as the smallest camera HFOV used so far to view each part of the environment. Because the PTZ camera center is fixed during a visual search event, this $2\frac{1}{2}$D, viewer-centered collection of cylindrical panoramic images is a sufficient representation of the local environment; a more complex 3D model is not necessary.

For $N_P = {}^{(\psi_{\max}-\psi_{\min})}/_{\triangle s}$, integer-valued pan coordinates, $P = \{1, \ldots, N_P\}$, index into a quantized pan space

$$[\psi_{\min}, \psi_{\min} + \triangle s),\ [\psi_{\min} + \triangle s, \psi_{\min} + 2\triangle s),$$
$$\ldots,\ [\psi_{\min} + (N_{\mathrm{P}} - 1)\,\triangle s, \psi_{\max}]\,.$$

And for $N_T = {}^{(\theta_{\max}-\theta_{\min})}/_{\triangle s}$, integer-valued tilt coordinates, $T = \{1, \ldots, N_T\}$, index into a quantized tilt space

$$[\theta_{\min}, \theta_{\min} + \triangle s)\,,\ [\theta_{\min} + \triangle s, \theta_{\min} + 2\triangle s)\,,$$
$$\ldots,\ [\theta_{\min} + (N_{\mathrm{T}} - 1)\,\triangle s, \theta_{\max}]\,.$$

The space of camera HFOVs is also quantized and indexed by integer-valued HFOV coordinates $Z = \{z_{\min}, z_{\min} + 1, z_{\min} + 2, \ldots, z_{\max}\}$ corresponding to actual camera HFOVs

$$\{z_{\min}\triangle s,\ (z_{\min} + 1)\,\triangle s,\ (z_{\min} + 2)\,\triangle s,\ \ldots,\ z_{\max}\triangle s\}$$

where $z_{\min} = {}^{\zeta_{\min}}/_{\triangle s}$ and $z_{\max} = {}^{\zeta_{\max}}/_{\triangle s}$. With the previous definitions, our pan, tilt, and HFOV coordinates all index into spaces uniformly sampled by $\triangle s$.

We assume that the local environment model $\mathbb{M}$ includes a color panoramic image, $\boldsymbol{C}$, a depth panoramic, $\boldsymbol{D}$, a target probability panoramic, $\boldsymbol{T}$, and an HFOV panoramic, $\boldsymbol{Z}$: $\mathbb{M} = \{\boldsymbol{C}, \boldsymbol{D}, \boldsymbol{T}, \boldsymbol{Z}\}$. For $(p,t) \in P \times T$, $\boldsymbol{C}[p,t]$, $\boldsymbol{D}[p,t]$, $\boldsymbol{T}[p,t]$, and $\boldsymbol{Z}[p,t]$ give the mean color, maximum depth, target probability, and smallest previously applied camera HFOV, respectively, for the $\triangle s \times \triangle s$ region of the environment indexed by (p,t). $\boldsymbol{C}$ comes directly from the PTZ camera color

images. The estimated depth, $\boldsymbol{D}$, may be generated from a known camera pose relative to a local terrain model, from an auxiliary depth sensor such as a LIDAR, or by applying any monocular depth estimation algorithm[24] to the color images. The target probability, $\boldsymbol{T}$, is obtained by applying any target detection algorithm (e.g., Bochkovskiy et al.,[2] Ren et al.,[25] Cetinkaya et al.,[26] and Tong and Wu[27]) to the color and depth images and then using learned performance statistics to estimate the probability that a real target is present given the detection confidence (Section 3.2.3). The HFOV panoramic, $\boldsymbol{Z}$, records the smallest HFOV used to view each part of the scene.

The panoramics in $\mathbb{M}$ are updated after each PTZ camera image is acquired. For simplicity, we approximate PTZ camera views as samples over rectangular regions in the space $P \times T$ of pan-tilt coordinates.* A PTZ view at position $(p, t, z) \in P \times T \times Z$ requires images in $\mathbb{M}$ to be updated at all cells (p', t') in a rectangular grid centered at (p, t) of width z and height $h = \rho \cdot z$, wherever $\boldsymbol{z} < \boldsymbol{Z}[p', t']$, where $\rho = {}^{n_r}/_{n_c}$ is the camera image aspect ratio. That is, a cell in the current view is updated only when the current view provides a higher resolution (smaller HFOV) image of that cell than any previous views. After updating $\mathbb{M}$, the visual search algorithm chooses, based on the current environment model and position of the camera, the next PTZ location to view, or ends the search process. An outline of a generic visual search algorithm is shown in Algorithm 1.

Algorithm 1 Basic visual search.

$\boldsymbol{v} \leftarrow (p_0, t_0, z_0)$
Initialize $\mathbb{M}$
repeat
  $\boldsymbol{v} \leftarrow$ ***NextView***$(\mathbb{M}, \boldsymbol{v})$
  if $\boldsymbol{v} \neq \emptyset$
    $\boldsymbol{I} \leftarrow$ ***CameraImage***$(\boldsymbol{v})$
    $\mathbb{M} \leftarrow$ ***Update***$(\mathbb{M}, \boldsymbol{I}, \boldsymbol{v})$
until $\boldsymbol{v} = \emptyset$

The path that a PTZ camera takes during a visual search event may be represented by a sequence $\boldsymbol{r} = ((p_i, t_i, z_i), i = 0, \ldots, n)$ of $|\boldsymbol{r}| = n + 1$ PTZ coordinates where $(p_i, t_i, z_i) \in P \times T \times Z$ and where (p_0, t_0, z_0) is the initial camera position. $\boldsymbol{r}$ defines the camera's path through the discretized 3D PTZ space as an ordered sequence

*In reality, a rectangular pinhole camera image maps to a distorted rectangle on a cylindrical panoramic, with increasing distortion for tilt angles farther from the orthogonal to the cylinder's vertical axes.

of PTZ destinations. The system iteratively moves the camera to each destination, acquires and processes a single image, updates its environment model, and then decides on the next PTZ position to view. The time needed to traverse this path and acquire and process images at all destinations is approximated as

$$t\left(\boldsymbol{r}\right) = \sum_{i=1}^{|\boldsymbol{r}|} \left(\sqrt{c_1 \triangle p^2 + c_2 \triangle t^2 + c_3 \triangle z^2} + c_4 \right) \tag{1}$$

where $\triangle p = p_{i-1} - p_i$, $\triangle t = t_{i-1} - t_i$, $\triangle z = z_{i-1} - z_i$, constants c_1, c_2, and c_3 determine the PTZ unit's pan, tilt, and zoom speeds, respectively, and c_4 is the time required to process a single camera image.

The *value* of a camera path $\boldsymbol{r}$ in an environment $\mathbb{E}$, denoted $V_{\mathbb{E}}\left(\boldsymbol{r}\right)$, is a function of the time required to execute the search, $t\left(\boldsymbol{r}\right)$, and the accuracy of predicted target locations as given by $\boldsymbol{T_f}$, the target probability panoramic produced in the final step of the search. The F-score[28] is commonly used to measure target detection accuracy:

$$F_{\mathbb{E}}\left(\boldsymbol{r}\right) = \frac{2 \times Precision(\boldsymbol{T_f}) \times Recall(\boldsymbol{T_f})}{Precision(\boldsymbol{T_f}) + Recall(\boldsymbol{T_f})}$$

where an appropriate threshold is applied to $\boldsymbol{T_f}$ to determine target positive and negative instances. Precision and recall, however, are typically not known for live, real environments; $F_{\mathbb{E}}\left(\boldsymbol{r}\right)$ may be used for algorithm training and evaluation when ground truth is known, but not for planning live visual searches in real environments. For the live, on-line visual search planning problem, the target location accuracy is replaced with the target uncertainty. This is computed as the sum over all pan-tilt positions in $P \times T$ of the target entropy:

$$\begin{aligned} H_{\mathbb{E}}\left(\boldsymbol{r}\right) \;\; = \;\; & -\sum_{p \in P} \sum_{t \in T} [\boldsymbol{T_f}[p,t] \log_2 \left(\boldsymbol{T_f}[p,t]\right) \\ & + (1 - \boldsymbol{T_f}[p,t]) \log_2 \left(1 - \boldsymbol{T_f}[p,t]\right)]. \end{aligned} \tag{2}$$

Then, the on-line visual search problem is to find a camera path $\boldsymbol{r}$ that maximizes the function

$$V_{\mathbb{E}}\left(\boldsymbol{r}\right) = -H_{\mathbb{E}}\left(\boldsymbol{r}\right) - \alpha t\left(\boldsymbol{r}\right) \tag{3}$$

where α is a constant scale factor.

Complexity of the Visual Search Problem

We note that the OP reduces to PTZSP, the optimization problem considered in this report. Loosely, this reduction can be established by (1) replacing each node from the OP with a target object (there will be no clutter objects), (2) assigning 2D pan-tilt coordinates to each target such that inter-target distances are a constant scale factor of the original OP inter-node distances, where the scale factor is chosen so that all inter-target distances are greater than $z_{\min}$, (3) using a detector that classifies an object with probability 1.0 if the camera HFOV is $z_{\min}$, and probability 0.5 otherwise, and (4) setting $c_1 = c_2 = 1$ and $c_3 = c_4 = 0$ in Eq. 1 (so path length is the 2D Euclidean distance). The optimal solution to the OP instance can be then be inferred from the optimal PTZ path. We are skipping the details of this routine conversion between these two optimal paths in the interest of space. A generic OP instance can be cast as a simplified version of the visual search problem considered in this report. As OP is Nondeterministic Polynomial-time (NP)-hard,[13] the Visual Search Problem considered in this report must also be NP-hard.

3.2 Environment Simulations

Our visual search algorithms have been developed and evaluated using a combination of two different urban simulators. The first is a procedurally generated, quasi-photorealistic,* 3D urban environment simulator that is used to learn a performance model of a pre-trained YOLO real-time object detection algorithm[2] on PTZ camera imagery of cluttered, large depth-of-field environments. Using this object detector performance model, a second, much faster, but non-photorealistic, 2D simulator of urban panoramic imagery is used to train and evaluate the proposed visual search algorithms. The 3D and 2D simulations, and the learned object detection models, are described in more detail in Sections 3.2.1, 3.2.2, and 3.2.3, respectively.

3.2.1 3D Environment Simulation

Each run of the 3D simulator generates a new random urban environment. The placement of objects into the environment is done in Python and rendering is

*By *quasi-photorealistic* we mean that all objects in the simulated environment are photorealisticly rendered, but that the complexity of the simulated environment may be less than that found in real-world environments.

performed using the open source Visualization Toolkit (VTK).[29] A single ground texture is placed onto a flat ground plane; roads are randomly laid out on a grid; buildings are placed near roads; fences and walls, plants, trees, animals, people, airborne objects, vehicles, and other miscellaneous objects are added in appropriate open spaces; some people are placed inside of buildings to look out of semitransparent windows; and finally, a sky with moving clouds and a background surrounding the entire region are added. Figures 1 and 2 show a few images of some 3D environments.

All objects in these 3D environments are created by texture mapping planar surfaces with cutouts from high-resolution digital photographs. This enables any 3D object to be easily inserted into our simulation given just a photograph of the object and a corresponding image with labeled object pixels. Some example texture cutouts are shown in Figure 3. For objects that are not naturally planar, such as people, cars, plants, and trees, the simulator rotates the cutouts about their vertical axes so that their normal vectors point toward the camera; This prevents unnatural foreshortening distortions and ensures these objects are photorealisticly rendered. Dynamic objects, such as people, animals, and airborne objects, can be set to move with semirandom linear motion, while plants and trees exhibit small oscillatory motions. In addition to color PTZ camera imagery, pixel-accurate depth and semantic labels are available for use as ground truth data. This approach to creating 3D urban models has one main disadvantage compared to true 3D models: all real-life nonplanar objects appear the same, up to scale, from any viewpoint. This problem is addressed by including photos of nonplaner objects taken from different viewpoints.

3.2.2 2D Environment Simulation

For a PTZ camera in a fixed location, properties of the 3D environment relevant to the visual search problem (such as depth from the camera, terrain class, and target presence) can be indexed using a pan-tilt coordinate system. The 2D urban simulator is therefore built around simulating panoramic images where column and row coordinates represent pan and tilt angles, respectively. In all of our experiments, panoramic images are 1800×350 in size corresponding to a 180° pan range, 35° tilt range, and a sample spacing of $\triangle s = 0.1^\circ$ in both pan and tilt angles. The value of $\triangle s$ is application dependent and roughly indicates the accuracy required of pan-tilt positions to accomplish the goals of the system. Let the camera image resolution,

Fig. 1 Images of simulated 3D urban environments: birds-eye view (top) and ground-view (bottom)

Fig. 2 More images of simulated 3D urban environments. Closeup of person (top) shows photorealism of simulation.

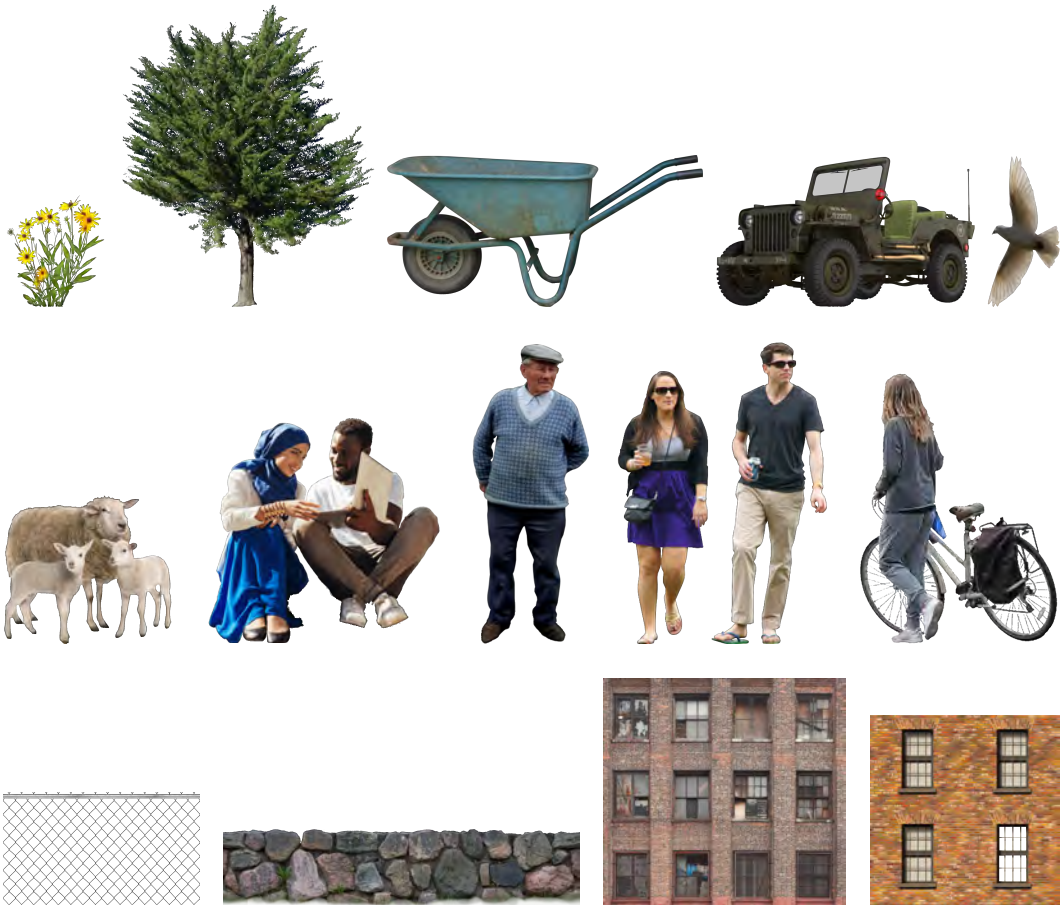


Fig. 3 Some planar texture cutouts used to create the 3D urban model (not to scale). Note that some windows in building textures are semi-transparent (e.g., the lower rightmost building texture), enabling partially obscured views of people inside of buildings.

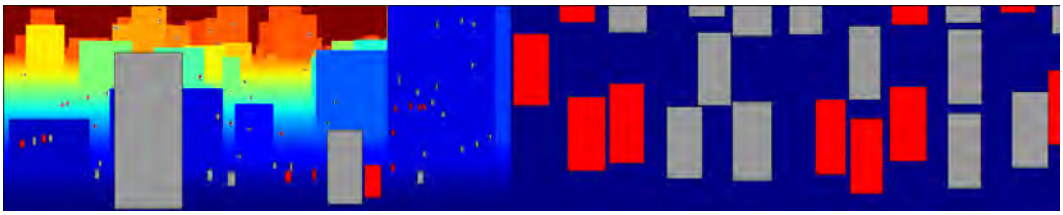


Fig. 4 A simulated panoramic image of an urban environment. Pixels are colored according to depth from the camera (blue is close, red is far), except red and gray rectangles, which represent targets and clutter, respectively. Objects on buildings simulate people looking out windows. Many targets and clutter exist at distant locations in the model but are not visible until the camera zooms in on those locations.

in pixels, be $n_r \times n_c$. Starting with a flat ground model and a PTZ camera whose vertical axis is a small random deviation from the orthogonal to the ground plane, an approximate model of an urban environment is created by projecting the ground plane depths into the panoramic image and then placing nonintersecting rectangles of constant depth at random locations in this image. The sizes of these rectangles are normally distributed with parameters dependent on the depth of the supporting surfaces (i.e., their distances from the camera) and the type of object represented by each rectangle: either building, target, or clutter.

In this report, "clutter" refers to any non-target object in the scene for which there is some value of camera zoom which, when used to view the object, results in a detection on that object with a positive confidence of the object being a true target. We need to explicitly represent clutter because a real object detector does not only respond to objects in its training set, but also to objects that it was not trained to detect, especially in parts of the scene imaged at low spatial resolution (e.g., distant regions). What is clutter depends on the object detector, not some inherent property of objects in the environment. If there was no clutter, any response from the object detector, no matter how weak, would have to be on a true target, and this would greatly eliminate the need to zoom in on objects to confirm their true classification (target or clutter).

Figure 4 shows an example of a randomly generated panoramic model of an urban environment. This 2D urban simulator does not attempt to model all nuances of real urban environments, but we feel these models are complex enough to evaluate PTZ visual search algorithms, and the ease of generating and using them outweighs their shortcomings.

3.2.3 Modeling the Object Detector

In order to use the 2D simulator to evaluate our algorithms, we must have a model of the YOLO object detector that can be applied to the simulated 2D scene in such a way that the model's behavior closely replicates the behavior of YOLO on real images of 3D objects. Our process for accomplishing this is described in this section.

The first step is to learn a mapping, given resolution and confidence arguments, to zoom- and depth-independent target probabilities. For any cell (p, t), we assume that we can obtain an estimate of the environment's depth, $d\,(p, t)$, within that cell. With this, we estimate the ground resolution of that cell when viewed with zoom z, denoted $r(p, t, z)$, in pixels per meter (ppm), which is the number of camera image pixels that one horizontal meter in the scene at depth $d\,(p, t)$ projects onto

$$r(p, t, z) = \frac{1}{d(p, t)\tan(z \triangle s / n_c)}.$$

The accuracy of image-based object detectors is often characterized in terms of this resolution parameter; this is related to "pixels on target," but is more broadly applicable to both target-like objects (targets and clutter) as well as background scenery. $r(p, t, z)$ is one of the arguments in the target probability map. The second argument is the target detector confidence.

The target probability mapping is learned by observing objects in a wide range of settings using the quasi-photorealistic 3D simulator. The process repeatedly generates random 3D environments, selects random camera placements in each, and then scans the environment with the PTZ camera set to a few different zooms. The YOLO object detector is applied to the set of generated images using a very low confidence threshold (0.01) to generate large numbers of detections. Ground truth imagery for all partially visible true targets is used to identify which of these detections are true positives or false positives, and also to augment with false negatives. Duplicate detections (with high Intersection Over Union [IoU]) are removed and a pan-tilt location is assigned to each object in this list, which we call the *key detections*. Then, the pan-tilt location of each key detection is viewed with the camera zoom set to 11 equally spaced values over the camera's full zoom range. The detector is applied to each of these images using a very low threshold, $c_{\min} = 0.01$, on the detection confidences, and then the detection confidence of

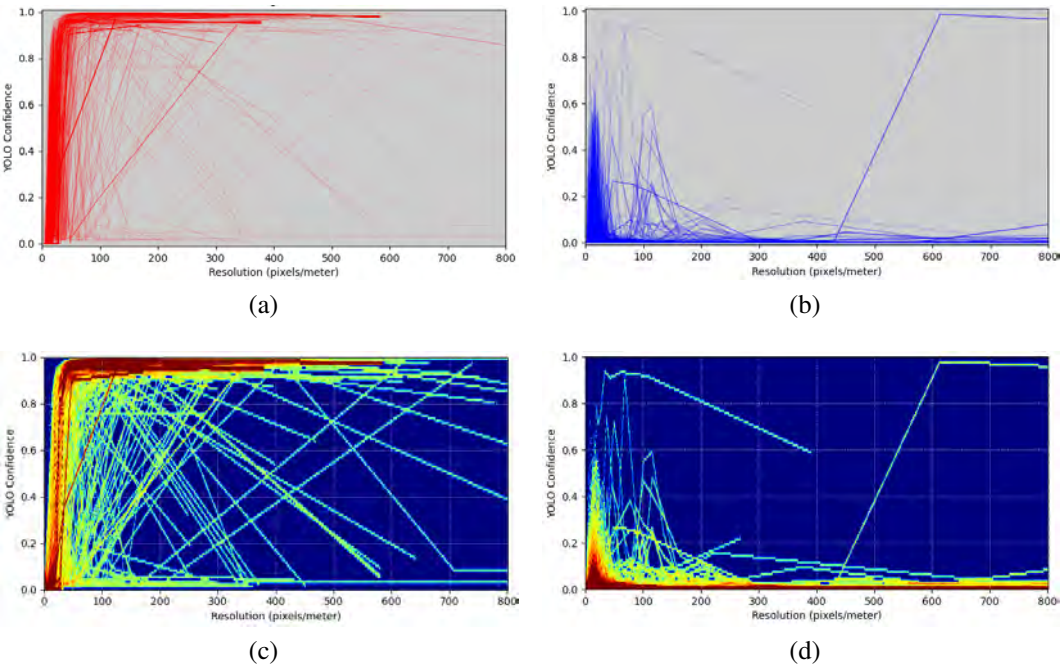


Fig. 5 YOLO performance model. (a) Interpolated YOLO confidence samples on true targets. (b) Interpolated YOLO confidence samples on clutter. (c) Histogram of interpolated target confidences on true targets. (d) Histogram of interpolated target confidences on clutter.

each key detection location is recorded along with the ground resolution (ppm). The 11 detector confidence values of each key detection are interpolated to produce a piecewise linear curve mapping ground resolution to detector confidence. Any missed true targets are assigned a detection confidence of zero. Figure 5 shows the generated detector confidence curves and histograms for true targets and clutter. The histograms $P(c, r \mid \mathcal{T})$ and $P(c, r \mid \mathcal{C})$ are discretized to 101 detector confidence values (c) in $[0, 1]$ and 801 ground resolution values (r) in $[0, 800]$ ppm, and then normalized to sum to one. In the following, we use $\mathcal{T}$ to represent the event that a target is present at the location in question, $\mathcal{C}$ the event that clutter is present, $\mathcal{B}$ the event that only background is present (neither clutter nor target), and $\mathcal{D}$ the event that there is a detection at that location. We see that YOLO produces mostly high-confidence (> 0.9) detections of true targets for ground resolutions of 50 ppm or higher, but there are many targets where this is not the case due to partial occlusion, over zoom on nearby targets, or objects with similar appearance to targets (e.g., statues of people). Conversely, detections on clutter are usually low confidence (< 0.1) for ground resolutions of 50 ppm or higher, but some clutter objects trigger high-confidence detections in small ranges of higher ground resolution.

To obtain real-world YOLO detection confidences in the 2D simulation, every target and clutter object in the 2D environment imitates one of the key detections described in the previous paragraph. 2D target objects imitate 3D targets by reproducing the same detection confidences as a function of ground resolution of the 2D object. Similarly, 2D clutter imitate the detector confidence curves of 3D clutter. Each 2D target is assigned a random target key detection to imitate, and each 2D clutter is assigned a random clutter key detection to imitate, where the probability of assignment of a particular 2D and 3D object is a zero-mean, normally distributed function of the difference in depths of the 2D and 3D objects. This ensures that 2D objects produce detector behavior similar to their paired 3D objects for similar zoom values.

When a visual search algorithm acquires a new PTZ image of the environment with the camera at (p, t, z), all locations $(p', t') \in [p - z/2 \leqq p' \leq p + z/2] \times [t - \rho z/2 \leqq t' \leq t + \rho z/2]$ in the model $\mathbb{M}$ for which $z < \boldsymbol{Z}[p', t']$ must be updated. The color, depth, and zoom images are updated by simply replacing the current color, depth, and zoom values by the new values. For the target probability image $\boldsymbol{T}$, the update depends on whether or not there is an object (clutter or target) at each location (p', t'). As described previously, objects in the 2D simulation imitate the detector confidence values of objects viewed in the 3D simulation. If there is an object $\mathcal{O}$ at location (p', t'), then $\boldsymbol{T}$ is updated with $\boldsymbol{T}[p', t'] \leftarrow P(\mathcal{T} \mid c, r)$ where $P(\mathcal{T} \mid c, r)$ is the probability of a target being present given a detection with confidence c at a location with ground resolution r. Here, c is the interpolated detector confidence of object $\mathcal{O}$ for resolution $r = r(p', t', z)$ and $P(\mathcal{T} \mid c, r)$ is calculated as

$$\begin{aligned} P(\mathcal{T} \mid c, r) &= \frac{P(\mathcal{T})P(c,r|\mathcal{T})}{P(c,r)} \\ &= \frac{P(\mathcal{T})P(c,r|\mathcal{T})}{P(\mathcal{T})P(c,r|\mathcal{T})+P(\mathcal{C})P(c,r|\mathcal{C})+P(\mathcal{B})P(c,r|\mathcal{B})} \end{aligned} \tag{4}$$

with $P(\mathcal{T}) = \rho_T$, $P(\mathcal{C}) = \rho_C$, and $P(\mathcal{B}) = \rho_{\mathcal{B}}$ where ρ_T, ρ_C, and $\rho_{\mathcal{B}}$ are 2D simulation parameters giving the expected relative target, clutter, and background densities, respectively, and satisfy $\rho_T + \rho_C + \rho_{\mathcal{B}} = 1$. By definition of *background*, $P(c, r \mid \mathcal{B}) = 0$ for any $c > 0$. Note that some objects will produce low confidence detections ($c < c_{\min}$) when viewed at certain resolutions, effectively giving no detection, but the target probability calculation in Eq. 4 is still valid.

We also need to estimate the probability that a target is present at a location (p', t')

in the case that the object detector does not detect anything in that location. Each of these locations has an associated ground resolution $r = r(p', t', z)$, but none will have a detector confidence value. The fact that there is no object at (p', t') is hidden from the search algorithms; the algorithms know only that the detector confidence at (p', t') was less than $c_{\min}$ and it is not known if this is due to the absence of a target or poor detector performance on an actual target (due possibly to a low resolution view). The probability of a target being present given there is no detection at resolution r is

$$P\left(\mathcal{T} \mid \bar{\mathcal{D}}, r\right) = \frac{P\left(\mathcal{T}, \bar{\mathcal{D}}, r\right)}{P\left(\bar{\mathcal{D}}, r\right)}$$

$$= \frac{P(\mathcal{T})P\left(\bar{\mathcal{D}}, r|\mathcal{T}\right)}{P(\mathcal{T})P\left(\bar{\mathcal{D}}, r|\mathcal{T}\right)+P(\mathcal{C})P\left(\bar{\mathcal{D}}, r|\mathcal{C}\right)+P(\mathcal{B})P\left(\bar{\mathcal{D}}, r|\mathcal{B}\right)}.$$

We make the substitutions

$$P\left(\bar{\mathcal{D}}, r|\ \mathcal{T}\right) = \frac{\sum_{0 \le c' < c_{\min}} P(c', r \mid \mathcal{T})}{\sum_{0 \le c' \le 1} P(c', r \mid \mathcal{T})},$$

$$P\left(\bar{\mathcal{D}}, r|\ \mathcal{C}\right) = \frac{\sum_{0 \le c' < c_{\min}} P(c', r \mid \mathcal{C})}{\sum_{0 \le c' \le 1} P(c', r \mid \mathcal{C})},$$

$$P\left(\bar{\mathcal{D}}, r|\ \mathcal{B}\right) = 1.$$

Then, updates to $\boldsymbol{T}$ at (p', t') are as follows:

$$\boldsymbol{T}[p', t'] \leftarrow \begin{cases} P\left(\mathcal{T} \mid c, r\right) & \text{if } \exists \mathcal{O} \text{ at } (p', t') \\ P\left(\mathcal{T} \mid \bar{\mathcal{D}}, r\right) & \text{otherwise} \end{cases} \tag{5}$$

3.3 Predicting Optimal Zoom

The value of a PTZ route (Eq. 3) depends on how well the route reduces target entropy in the local environment model (due to acquisition of informative images), as well as the time taken to acquire and process those images. Hence, we wish to use the minimum number of camera images necessary to sufficiently reduce this entropy. The environment model's target entropy at (p, t) is considered sufficiently low when $\boldsymbol{T}[p, t] \le 1 - t_{\text{conf}}$ or $\boldsymbol{T}[p, t] \ge t_{\text{conf}}$. An optimal zoom for a point in the environment, considering each point as independent, is estimated as the largest

zoom (HFOV) that provides the required reduction in target entropy. We therefore estimate the camera zoom, as a function of scene depth, needed to produce low-entropy object detections. The expected minimum ground resolution needed to ensure $\boldsymbol{T}[p,t] \leq 1 - t_{\text{conf}}$ is

$$r_{\text{conf}} = \min_{r} \left| P\left(\mathcal{T} \mid \bar{\mathcal{D}}, r\right) - (1 - t_{\text{conf}}) \right|.$$

For a scene point at (p,t) with estimated depth $d = D[p,t]$, the maximum camera zoom (HFOV) at which we expect this point needs to be viewed to ensure its ground resolution is at least r_{conf} is

$$z(d) = \frac{n_c \arctan\left(1/d\right)}{r_{\text{conf}}} \tag{6}$$

where n_c is the number of columns in the camera images, and due to camera physical zoom limitations, $z(d) \geq z_{\min}$. Viewing a scene point at depth d with zoom $z(d)$ is expected to produce the required entropy reduction, but it is not guaranteed due to the random nature of object appearances and behavior of the object detector.

All of our visual search algorithms begin by performing a quick scan of the environment ROI (via Algorithm 2, ***QuickScan***) to initialize depth estimates for each cell in the environment model $\mathbb{M}$. With these depth estimates, we augment the environment model $\mathbb{M}$ with a panoramic image $\bar{\boldsymbol{Z}}$ indicating the expected optimal zoom at each model cell: $\bar{\boldsymbol{Z}}[p,t] = z\left(\boldsymbol{D}[p,t]\right)$. The augmented environment model is $\mathbb{M} = \left\{\boldsymbol{C}, \boldsymbol{D}, \boldsymbol{T}, \boldsymbol{Z}, \bar{\boldsymbol{Z}}\right\}$.

3.4 Quadtree Decomposition

The quadtree decomposition algorithm, ***QuadtreeDecomp***, shown in Algorithm 3, starts off with a "quick scan" of the environment via algorithm ***QuickScan***, scanning the entire ROI from left to right, bottom to top, using the largest possible camera HFOV, and updating the environment model after acquiring each image. Algorithm ***View***, shown in Algorithm 4, is responsible for moving the camera, in simulation or in a real-world system, and then updates the environment model. ***QuadtreeDecomp*** then examines the current model target probabilities associated with each of these initial views to determine if any are ambiguous (i.e., $1 - t_{\text{conf}} < \boldsymbol{T}[p,t] < t_{\text{conf}}$). This is done by algorithm ***ExploreRect***, shown in Algorithm 5, by splitting each view into four quadrants and any quadrant containing a cell (p,t) with

Algorithm 2 ***QuickScan*** (m_p, m_t)

Perform quick, back and forth, bottom to top, scan of environment using largest camera HFOV. $m_p \times m_t$ *are dimensions of environment model to initialize. Return the environment model* $\mathbb{M}$*.*

$\mathbb{M} \leftarrow \{\boldsymbol{Z}[\cdot,\cdot] \leftarrow \infty;\ \boldsymbol{T}[\cdot,\cdot] \leftarrow 1\}$
$z \leftarrow z_{\max}$ # HFOV
$h \leftarrow \rho \cdot z$ # VFOV
$N_p \leftarrow \lceil m_p/z \rceil$; $N_t \leftarrow \lceil m_t/h \rceil$
$\triangle p \leftarrow (m_p - z)/(N_p - 1)$
$\triangle t \leftarrow (m_t - h)/(N_t - 1)$
$p \leftarrow z/2$
$t \leftarrow m_t - h/2$
for $j \leftarrow 1, 2, \ldots, N_t$ **do**
  for $k \leftarrow 1, 2, \ldots, N_p$ **do**
    $\mathbb{M} \leftarrow$ ***View*** $(p, t, z, \mathbb{M})$
    $p \leftarrow p + \triangle p$ # pan left or right
  end for
  $p \leftarrow p - \triangle p$ # stay at last pan
  $t \leftarrow t - \triangle t$ # tilt up
  $\triangle p \leftarrow -\triangle p$ # reverse pan direction
end for
return $\mathbb{M}$

ambiguous target probability is viewed with camera zoom set to half the previous HFOV. $t_{\text{conf}} \in [0, 1]$ is a system parameter that identifies the target probability at which there is sufficiently low entropy in the presence of a target. (We typically set t_{conf} in the range $[0.8, 0.9]$.) This decomposition process repeats until all cells (p, t) in the viewed rectangles are such that $\boldsymbol{T}[p, t] \leq 1 - t_{\text{conf}}$ or $\boldsymbol{T}[p, t] \geq t_{\text{conf}}$ or the minimum HFOV is reached. Note that this approach optimizes a discrete version of the target entropy (Eq. 2). In the algorithm descriptions that follow, we use the notation ***rect(p,t,z)*** to represent the set of integer coordinates contained in a rectangle centered at (p, t) with size $z \times (\rho \cdot z)$ (width$\times$height) where ρ is the camera image aspect ratio.

Algorithm 3 ***QuadtreeDecomp*** (m_p, m_t)

Quadtree decomposition search of environment with pan, tilt dimensions $m_p \times m_t$.

$z \leftarrow z_{\max}$ # HFOV
$h \leftarrow \rho \cdot z$ # VFOV
$N_p \leftarrow \lceil m_p/z \rceil$; $N_t \leftarrow \lceil m_t/h \rceil$
$\triangle p \leftarrow (m_p - z)/(N_p - 1)$
$\triangle t \leftarrow (m_t - h)/(N_t - 1)$
$\mathbb{M} \leftarrow$ ***QuickScan*** (m_p, m_t) # initialize $\mathbb{M}$
$p \leftarrow z/2$
$t \leftarrow m_t - h/2$
for $j \leftarrow 1, 2, \ldots, N_t$ **do**
  for $k \leftarrow 1, 2, \ldots, N_p$ **do**
    $\mathbb{M} \leftarrow$ ***ExploreRect*** $(p, t, z, \mathbb{M})$
    $p \leftarrow p + \triangle p$ # pan left or right
  end for
  $p \leftarrow p - \triangle p$ # stay at last pan
  $t \leftarrow t - \triangle t$ # tilt up
  $\triangle p \leftarrow -\triangle p$ # reverse pan direction
end for

Algorithm 4 ***View*** $(p, t, z, \mathbb{M})$

View the environment with an image centered at pan, tilt coordinate (p, t) *and HFOV* z*. Then, update the environment model* $\mathbb{M}$*. The environment* $\mathbb{E}$ *is hidden from the search algorithm in real-world systems.*

$h \leftarrow \rho \cdot z$ # VFOV
for $p - z/2 \leqq p' \leq p + z/2$
  for $t - h/2 \leqq t' \leq t + h/2$ **do**
    if $z < \boldsymbol{Z}[p', t']$ **then**
      $\boldsymbol{Z}[p', t'] = z$
      if $\exists$ object $\in \mathbb{E}$ at(p', t') **then**
        $\boldsymbol{T}[p', t'] \leftarrow P(\mathcal{T} \mid c, r)$
      else
        $\boldsymbol{T}[p', t'] \leftarrow P\left(\mathcal{T} \mid \bar{\mathcal{D}}, r\right)$
      end if
    end if
  end for
end for
return $\mathbb{M}$

Algorithm 5 ***ExploreRect*** $(p, t, z, \mathbb{M})$

Explore in greater depth the rectangle centered at pan, tilt coordinate (p, t) *and HFOV* z*.*

if $z/2 < z_{\min}$ **then**
  return # can't zoom-in any further
end if
$h \leftarrow \rho \cdot z$ # VFOV
for $\triangle p \in \{-z/4, z/4\}$ **do**
  for $\triangle t \in \{-h/4, h/4\}$ **do**
    if $\exists\,(p', t') \in$ ***rect*** $(p + \triangle p, t + \triangle t, z/2)$
        $\mid 1 - t_{\text{conf}} < \boldsymbol{T}[p', t'] < t_{\text{conf}}$ **then**
      $\mathbb{M} \leftarrow$ ***View*** $(p + \triangle p, t + \triangle t, z/2, \mathbb{M})$
      $\mathbb{M} \leftarrow$ ***ExploreRect*** $(p + \triangle p, t + \triangle t, z/2, \mathbb{M})$
    end if
  end for
end for
return $\mathbb{M}$

3.5 Genetic Algorithms

Genetic algorithms (GAs)[30] perform search and optimization loosely based on a model of natural selection and genetics. A population of encoded solutions is evolved over many generations using selection, mutation, and crossover operators, where the likelihood that any individual solution participates in one of these operations and generates offspring in the next generation is related to the fitness of the parent's solution. Two GAs have been developed for the PTZ search problem: ***StaticGA*** and ***DynamicGA***, shown in Algorithms 6 and 7, respectively. In ***StaticGA***, the environment model $\mathbb{M}$ is static during the GA planning process, while in ***DynamicGA***, the environment model $\mathbb{M}$ is dynamic during this process. Both algorithms start by performing the same "quick scan" of the ROI that ***QuadtreeDecomp*** performs. ***StaticGA*** then uses ***GASearch***, Algorithm 8, to plan a PTZ route for the entire ROI based on the environment model $\mathbb{M}$ produced by ***QuickScan***; unlike ***QuadtreeDecomp***, new observations from camera views acquired while executing this PTZ route do not change the planned route, even when ambiguities remain. ***DynamicGA*** uses ***GASearch*** to plan an initial PTZ route, but like ***QuadtreeDecomp***, as the route is executed, the route may be updated (using ***GASearch***) based on observations in newly acquired images. Additional camera views will be acquired, when possible, wherever target probabilities remain ambiguous after viewing the environment at a zoom previously expected to resolve

Algorithm 6 ***StaticGA*** $(m_p, m_t, n_{pop}, n_{gen}, \tau_{max}, s_{max})$

Static GA search of environment with pan, tilt dimensions $m_p \times m_t$*, population size* n_{pop}*, number of generations* n_{gen}*, maximum PTZ time* $\tau_{\max}$*, and maximum number of PTZ steps* s_{max}*.* (p_0, t_0, z_0) is the starting camera position.

$\mathbb{M} \leftarrow$ ***QuickScan*** (m_p, m_t) # initialize $\mathbb{M}$
$\mathbb{M} \leftarrow$***DepthToZoom*** $(\mathbb{M})$ # add $\bar{\boldsymbol{Z}}$, Eq. 6
$\boldsymbol{p} \leftarrow$ ***InitPop*** $(n_{pop}, s_{max}, p_0, t_0, z_0)$
$\boldsymbol{p}, \mathbb{M}, b \leftarrow$ ***GASearch*** $(m_p, m_t, n_{gen}, \tau_{max}, s_{max}, \boldsymbol{p}, \mathbb{M})$
for $k \leftarrow 1$ **to** $|\boldsymbol{p}_b|$ **do**
  $p, t, z \leftarrow \boldsymbol{p}_b^k$
  $\mathbb{M} \leftarrow$ ***View*** $(p, t, z, \mathbb{M})$
end for

those ambiguities.

3.5.1 Initial Population

In the GAs, each individual in the population is a proposed solution to the PTZ search problem. ***InitPop***, shown in Algorithm 9, creates an initial population as a sequence $\boldsymbol{p} = \left(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{n_{\text{pop}}}\right)$ of n_{pop} individuals where each individual $\boldsymbol{p}_i$ is initialized to a sequence of $s_{\max}$ random PTZ values: $\boldsymbol{p}_i = (\boldsymbol{p}_i^0, \boldsymbol{p}_i^1, \ldots, \boldsymbol{p}_i^{s_{\max}})$ where $s_{\max}$ is the maximum number of steps (viewed PTZ images) in any solution, $\boldsymbol{p}_i^k \in P \times T \times Z$ for all k, and $\boldsymbol{p}_i^0 = (p_0, t_0, z_0)$ is the starting position of the camera, which is a constant for all solutions. For notational convenience, we will reference the individual pan, tilt, and zoom values of any 3-tuple $\boldsymbol{p}_i^k$ by $\psi(\boldsymbol{p}_i^k)$, $\theta(\boldsymbol{p}_i^k)$, and $\zeta(\boldsymbol{p}_i^k)$, respectively, and define the function $\boldsymbol{RandInt}(S_1, S_2, ..., S_n)$ to return an *n*-tuple of integers where the k^{th} element is a uniformly distributed random element of the set S_k.

3.5.2 Fitness Function

As shown in Eq. 3, we would like to assign a fitness to every solution based on how well the solution is expected to reduce target entropy in the environment model (i.e., information gain), and on the time taken to acquire and process its images. The actual information gain is not known until the PTZ route is executed, and efficiently predicting it is challenging.[31] The time to acquire and process a sequence of PTZ images, however, can be accurately predicted. Instead of trying to predict the information gain of a solution, we use an easily computed measure of how much of the scene will be viewed at the expected optimal zoom, as defined by $\bar{\boldsymbol{Z}}$. The fitness

Algorithm 7 ***DynamicGA*** $(m_p, m_t, n_{pop}, n_{gen}, \tau_{max}, s_{max})$

Dynamic GA search of environment with pan, tilt dimensions $m_p \times m_t$*, population size* n_{pop}*, number of generations* n_{gen}*, maximum PTZ time* τ_{max}*, and maximum number of PTZ steps* s_{max}*.* (p_0, t_0, z_0) is the starting camera position.

$\mathbb{M} \leftarrow$ ***QuickScan*** (m_p, m_t) # initialize $\mathbb{M}$
$\mathbb{M} \leftarrow$ ***DepthToZoom*** $(\mathbb{M})$ # add $\bar{\boldsymbol{Z}}$, Eq. 6
$\boldsymbol{p} \leftarrow$ ***InitPop*** $(n_{pop}, s_{max}, p_0, t_0, z_0)$
$\boldsymbol{p}, \mathbb{M}, b \leftarrow$ ***GASearch*** $(m_p, m_t, n_{gen}, \tau_{max}, s_{max}, \boldsymbol{p}, \mathbb{M})$
$n_{gen} \leftarrow n_{gen}/4$ # do quicker route updates
while $b > 0$ **do**
  $p, t, z \leftarrow \boldsymbol{p}_b^1$ # first step in best route
  $\mathbb{M} \leftarrow$ ***View*** $(p, t, z, \mathbb{M})$
  for $(p', t') \in$ ***rect*** (p, t, z) **do**
    if $z \leq \bar{\boldsymbol{Z}}[p', t']$
      if $1 - t_{\text{conf}} < \boldsymbol{T}[p', t'] < t_{\text{conf}}$
        $\bar{\boldsymbol{Z}}[p', t'] \leftarrow \max(\bar{\boldsymbol{Z}}[p', t']/2, z_{\min})$ # zoom more here
      else
        $\bar{\boldsymbol{Z}}[p', t'] \leftarrow \infty$ # don't look here again
      end if
    end if
  end for
  for $i \leftarrow 1$ **to** n_{pop} **do** # set new starting point in all solutions
    $\boldsymbol{p}_i \leftarrow$ ***LeftShift*** $(\boldsymbol{p}_i)$
    $\boldsymbol{p}_i^0 \leftarrow (p, t, z)$
    $\boldsymbol{p}_i^{s_{\max}} \leftarrow$ ***RandInt*** (P, T, Z)
  end for
  $\boldsymbol{p}, \mathbb{M}, b \leftarrow$ ***GASearch*** $(m_p, m_t, n_{gen}, \tau_{max}, s_{max}, \boldsymbol{p}, \mathbb{M})$
end while

Algorithm 8 ***GASearch***$(\boldsymbol{m_p}, \boldsymbol{m_t}, \boldsymbol{n_{pop}}, \boldsymbol{n_{gen}}, \boldsymbol{\tau_{max}}, \boldsymbol{s_{max}}, \boldsymbol{p}, \mathbb{M})$

GA search of environment with pan, tilt dimensions $m_p \times m_t$, initial population $\boldsymbol{p}$ of size n_{pop}, for number of generations n_{gen}, maximum PTZ time τ_{max}, maximum number of PTZ steps s_{max}, and environment model $\mathbb{M} = \{\boldsymbol{C}, \boldsymbol{D}, \boldsymbol{T}, \boldsymbol{Z}, \bar{\boldsymbol{Z}}\}$. Returns updated population $\boldsymbol{p}$, updated model $\mathbb{M}$, and index b of best solution.

if $\min(\bar{\boldsymbol{Z}}) = \infty$
  $b \leftarrow 0$ # nowhere else to look
else
  $c_{\mathrm{n}} \leftarrow \lceil \log_2(n_{\mathrm{pop}})/2 \rceil$ # num. of EDGCA clusters
  $c_{\mathrm{s}} \leftarrow \lceil n_{\mathrm{pop}}/c_{\mathrm{n}} \rceil$ # size of each cluster
  for $k \leftarrow 1$ **to** $n_{\mathrm{gen}} + 1$ **do**
    $\omega_{\max} \leftarrow 0$ # fitness of best solution
    for $i \leftarrow 1$ **to** n_{pop} **do** # get fitness of all $\boldsymbol{p}_i$
      $\omega_i, \kappa_i \leftarrow$ ***Fitness***$(\tau_{max}, s_{max}, \boldsymbol{p}_i, \mathbb{M})$
      if $\omega_i > \omega_{\max}$ # save best solution
        $\omega_{\max} \leftarrow \omega_i$
        $b \leftarrow i$
      end if
    end for
    if $k = n_{\mathrm{gen}} + 1$
      break # return best solution
    end if
    do EDGCA selection and crossover on $\boldsymbol{p}$ with
      c_{n} clusters of size c_{s} (Section 3.5.3)
    do mutations (Section 3.5.4)
  end for
end if
return $\boldsymbol{p}, \mathbb{M}, b$

Algorithm 9 ***InitPop***$(n_{pop}, s_{max}, p_0, t_0, z_0)$

Initialize the GA population. (p_0, t_0, z_0) is the starting camera position. s_{max} is the maximum number of steps (camera images) in a solution. See Section 3.1 for definitions of P, T and Z.

for $i \leftarrow 1$ **to** n_{pop} **do**
  $\boldsymbol{p}_i^0 = (p_0, t_0, z_0)$
  for $k \leftarrow 1$ **to** s_{max} **do**
    $\boldsymbol{p}_i^k \leftarrow$ ***RandInt***(P, T, Z)
  end for
end for
return $\boldsymbol{p}$

of an individual $\boldsymbol{p}_i$ is then

$$f\left(\boldsymbol{p}_i\right) = \max_{1 \le k \le s_{\max}} \Big[\sum_{(p,t) \in P \times T} \mathbf{1}_{\mathrm{T}} \left(\hat{\boldsymbol{Z}}^k_{\boldsymbol{p}_i}\left[p,t\right] \le \bar{\boldsymbol{Z}}[p,t] \right) - \alpha_{\mathrm{d}} \sum_{1 \le j \le k} \left\| \boldsymbol{p}_i^j - \boldsymbol{p}_i^{j-1} \right\| - \alpha_{\mathrm{s}} k \Big] \tag{7}$$

where $\mathbf{1}_T$ is the indicator function that maps the value *True* to 1 and *False* to 0, $\hat{\boldsymbol{Z}}^k_{\boldsymbol{p}_i}\left[p,t\right]$ is the smallest camera zoom at which model cell (p,t) is viewed in steps 1 through k of $\boldsymbol{p}_i$:

$$\hat{\boldsymbol{Z}}^k_{\boldsymbol{p}_i}\left[p,t\right] = \min_{0 \le j \le k} \{\zeta(\boldsymbol{p}_i^j) \mid (p,t) \in rect(\boldsymbol{p}_i^j)\},$$

and α_{d} and α_{s} are constant scale factors. The first term in the maximization of Eq. 7 counts the number of pan-tilt cells in the model that are "covered" (viewed at or below the expected optimal zoom) by the first k steps of $\boldsymbol{p}_i$. This is an estimate of how well the search will detect true targets and suppress clutter in the region of interest. The second term is the scaled distance traveled (or time to move) in PTZ space of the first k steps of $\boldsymbol{p}_i$. The third term is the scaled length (number of images) of $\boldsymbol{p}_i$. The length of $\boldsymbol{p}_i$ is the number of images of $\boldsymbol{p}_i$ that should be acquired and processed if $\boldsymbol{p}_i$ is selected as the best solution and is the value of k in Eq. 7 at which the maximum occurs:

$$\left|\boldsymbol{p}_i\right| = \arg\max_{1 \le k \le s_{\max}} \Big[\sum_{(p,t) \in P \times T} \mathbf{1}_{\mathrm{T}} \left(\hat{\boldsymbol{Z}}^k_{\boldsymbol{p}_i}\left[p,t\right] \le \bar{\boldsymbol{Z}}[p,t] \right) - \alpha_{\mathrm{d}} \sum_{1 \le j \le k} \left\| \boldsymbol{p}_i^{j-1} - \boldsymbol{p}_i^{j} \right\| - \alpha_{\mathrm{s}} k \Big].$$

So, the parameters α_{d} and α_{s} may be thought of as scaling the time required to acquire and process images in the search relative to the expected quality of the search result. Computation of a solution's fitness is perfromed by ***Fitness***, shown in Algorithm 10.

Algorithm 10 ***Fitness*** $(\boldsymbol{\tau_{max}}, \boldsymbol{s_{max}}, \boldsymbol{p}, \mathbb{M})$

Compute the fitness of one individual $\boldsymbol{p}$*. Return fitness,* ω*, of* $\boldsymbol{p}$ *and index,* κ*, of position giving maximum fitness.*

$\boldsymbol{Z}' \leftarrow \mathbf{1}_{\mathrm{T}} \left(\boldsymbol{Z} > \bar{\boldsymbol{Z}} \right)$ # deeper views needed here
$\tau \leftarrow 0$ # time to move camera
$\omega \leftarrow 0$ # current fitness
$\omega_{\max} \leftarrow 0$ # max fitness
for k = 1 **to** $s_{\max}$ **do**
  $\Delta\tau \leftarrow \alpha_{\mathrm{d}} \left\| \boldsymbol{p}^k - \boldsymbol{p}^{k-1} \right\|$
  $\tau \leftarrow \tau + \Delta\tau$
  if $\tau > \tau_{\max}$
    break # max PTZ time exceeded
  end if
  $\omega \leftarrow \omega - \Delta\tau - \alpha_{\mathrm{s}}$
  for $(p, t) \in$ ***rect***$(\psi(\boldsymbol{p}^k), \theta(\boldsymbol{p}^k), \zeta(\boldsymbol{p}^k))$ **do**
    if $\zeta(\boldsymbol{p}^k) \leq \bar{\boldsymbol{Z}}[p, t]$ **and** $\boldsymbol{Z}'[p, t] = 1$
      $\omega \leftarrow \omega + 1$
      $\boldsymbol{Z}'[p, t] \leftarrow 0$ # no more reward at (p, t)
    end if
  end for
  if $\omega > \omega_{\max}$
    $\omega_{\max} \leftarrow \omega$
    $\kappa \leftarrow k$
  end for
end for
return ω, κ

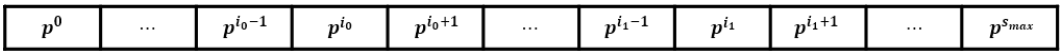
3.5.3 Selection and Crossover Operations

The selection operation is used to choose which candidate solutions may pass their attributes (genes) on to the next generation of candidate solutions. Individuals with higher fitness are more likely to be selected to reproduce in this process. Two selected (parent) solutions from the current generation reproduce through crossover, where parts of the two parents are combined to form the child, and then mutation, where individual genes in the child may be altered.

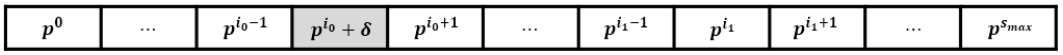
GAs sometimes suffer from premature convergence to local optima due to selection and crossover operations allowing highly fit individuals to dominate the population and therefore reduce the genetic diversity of the population.[32] We use the Elitist and Dynamic Genetic Clustering Algorithm (EDGCA) proposed in Malik and Wadhwa[33] to help reduce premature convergence. This approach first evaluates the fitness of each candidate solution and then splits the population into a fixed number of random, fixed-sized clusters. Within each cluster, single-point crossover is performed on the fittest solution from the first half of the cluster with the fittest solution from the second half. Of these three candidate solutions, the two parents and the child, the fittest is moved into the next generation. Finally, all remaining members of the cluster are replaced with mutated (Section 3.5.4) versions of the fittest individual from the crossover step.

3.5.4 Mutation Operations

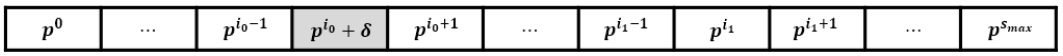
Mutations are generally small random changes to candidate solutions that produce new solutions. These are used to maintain and introduce diversity into a population. Below, we describe the nine types of mutations used by our GAs. Because the first position in all candidate solutions represents the current position of the PTZ camera, whose image has already been acquired and processed, this position can't be changed, so this position will never be mutated. Figures 6 and 7 show the nine mutations, labeled as $M1, \ldots, M9$, used by our GAs. These are randomly applied to the crossover result of each cluster (Section 3.5.3) with probabilities $m_p^i, i = 1, \ldots, 9$, respectively, where $\sum_{i=1}^{9} m_p^i = 1$.



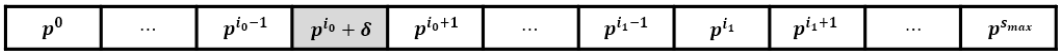
(a) Candidate solution prior to applying a mutation.



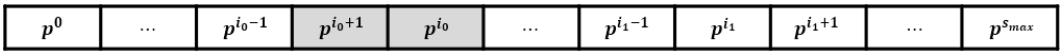
(b) *M1*: Small uniform perturbation of a single PTZ view. $\boldsymbol{\delta} \in \boldsymbol{RandInt}(-20\colon 20, -20\colon 20, 1\colon 1)$.



(c) *M2*: Large uniform perturbation of a single PTZ view. $\boldsymbol{\delta} \in \boldsymbol{RandInt}(P, T, Z)$.

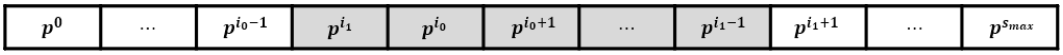


(d) *M3*: Random normal perturbation of a single PTZ view. $\boldsymbol{\delta} \sim \boldsymbol{N}(\mu, \sigma)$ where $\mu = (0, 0, 0)$ and $\sigma = (500, 200, 0.25)$.



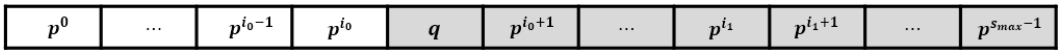
(e) *M4*: Swap two adjacent PTZ views.

(f) *M5*: Reverse a PTZ subpath.

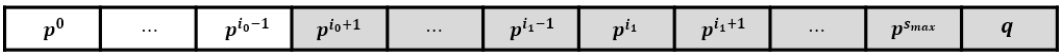


(g) *M6*: Rotate a PTZ subpath left or right. Rotate right shown.

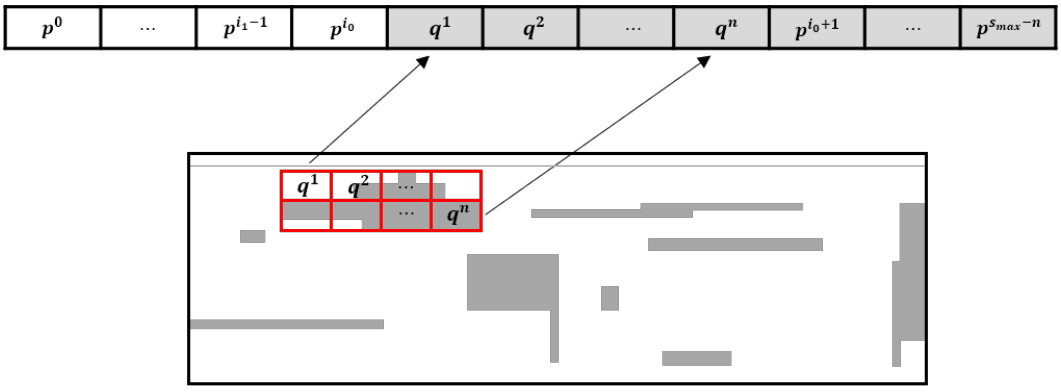
Fig. 6 GA mutations. The representation of a candidate solution prior to performing a mutation is shown in (a). Each p^i is a PTZ 3-tuple $(p, t, z) \in P \times T \times Z$. (b)–(g) show the candidate solution after performing each type of mutation. In these figures, the gray elements of a candidate solution are those that are changed as a result of the mutation operation. To identify where in the candidate solution a mutation occurs, one or two indices, i_0 and i_1, are needed depending on the type of mutation. These are random integers where $1 < i_0 \leq s_{max}$ for single-point mutations and $1 < i_0 < i_1 \leq s_{max}$ for subpath mutations. All mutations involving random perturbations of PTZ points p by $\boldsymbol{\delta}$ are cropped so that $p + \boldsymbol{\delta} \in P \times T \times Z$.



(a) *M7*: Insert a new PTZ view $\boldsymbol{q}$ at a random position. $\boldsymbol{q} \in \boldsymbol{RandInt}(P, T, Z)$.



(b) *M8*: Delete a random existing PTZ view and add a new random view $\boldsymbol{q}$ to the end. $\boldsymbol{q} \in \boldsymbol{RandInt}(P, T, Z)$.



(c) *M9*: Insert new PTZ views covering a randomly chosen connected component of currently non-covered cells. The "not-covered" image is the image $\mathbf{1}_{\mathrm{T}}\left(\hat{\boldsymbol{Z}}_{\boldsymbol{p}}^{k^*}[p,t] > \bar{\boldsymbol{Z}}[p,t]\right)$ of pan-tilt cells in the model that are not covered by the first k^* steps of the candidate solution $\boldsymbol{p}$. That is, the gray cells in this image are viewed by $\boldsymbol{p}$ with a zoom that is larger than the expected optimal zoom. Here, k^* is the number of steps that maximizes the evaluation of the fitness of $\boldsymbol{p}$. The connected components in this image (the gray blobs) are computed (and saved) for each candidate solution each time its fitness is evaluated. For the selected connected component, the smallest optimal zoom $\bar{\boldsymbol{Z}}[p,t]$ in that connected component is used to uniformly tessellate (as shown by the red rectangles) its bounding rectangle into PTZ subviews $\boldsymbol{q}^1, \ldots, \boldsymbol{q}^n$, and these subviews are inserted into the candidate solution to the right of the $\boldsymbol{p}^{i_0}$, which minimizes $\left\|\boldsymbol{q}^{\lfloor n/2 \rfloor} - \boldsymbol{p}^{i_0}\right\|$. All $\boldsymbol{p}^j$ with $j > i_0$ are shifted right to make room for $\boldsymbol{q}^1, \ldots, \boldsymbol{q}^n$.

Fig. 7 Additional GA mutations

3.5.5 Static and Dynamic Genetic Algorithms

The static and dynamic versions of the GA visual search algorithms, ***StaticGA*** and ***DynamicGA***, both start off with a quick scan of the environment, followed by initialization of the environment model and a population of candidate solutions, and then call ***GASearch*** which searches for the best PTZ path with which to scan the environment. After this point, the two algorithms differ. ***StaticGA*** executes the exact PTZ path returned by ***GASearch***; no adaptations are made to the path due to information gained from viewing PTZ locations in this path. In contrast, ***DynamicGA***, after viewing a PTZ location from the current best solution, updates the environment model $\mathbb{M}$, resets the starting point of all candidate solutions to the current PTZ location, and then calls ***GASearch*** to find a new best solution given the updated model. These latter calls to ***GASearch*** are performed using only a fraction of the number of generations as used in the original call because, assuming changes to $\mathbb{M}$ are small, members in the current population should already be close to the optimal solution for the updated model. This process is repeated until the updated best solution is empty, meaning that no additional views of the environment are needed. In addition to the environment model updates performed by ***View***, ***DynamicGA*** updates after every view the predicted optimal zoom image, $\bar{\boldsymbol{Z}}$, at all pan-tilt locations that were viewed at the predicted optimal zoom but whose target probability values remain ambiguous. The new predicted optimal zoom at these locations is simply set to half the previous value.

3.6 TSP Search with Lazy Constraints

Algorithms that solve the TSP[34] identify the shortest route between a set of locations that must be visited. This NP-complete problem is typically posed as an instance of a Binary Integer Linear Program (BILP), where a linear cost function is minimized subject to a set of linear constraints on a set of binary-valued decision variables. The execution time of the BILP-formulation can be unattractive due to the size of the subtour elimination constraints, which is exponentially related to the number of locations. This issue can be handled by an iterative approach that starts with a set of constraints that do not include any subtour elimination constraints. If the resulting solution contains a subtour, which can be identified in linear time, it is forbidden by the addition of an appropriate linear constraint, and the new TSP instance is solved anew. This iterative procedure, known as TSP with lazy constraints, will eventually yield the optimal solution to the original TSP instance, with significant

improvements in execution time (cf. Pferschy and Stanek,[35] for example). The TSP-instances in this report were solved using *Gurobi*,[36] which supports lazy constraint generation.

We use the TSP with lazy constraints procedure in algorithm ***LazyTSPPath***, shown in Algorithm 11, to choose PTZ paths for the visual search problem. The main ideas behind this algorithm are illustrated in Figure 8. Initially, the rectangular, high-ambiguity regions in the panoramic image that need further exploration are identified (Tier 1). Each rectangle corresponds to a (p_i^1, t_i^1, z_i^1)-coordinate. The superscript refers to the Tier, and the subscript refers to the rectangular region within the Tier. The shortest path that connects these (p, t, z)-coordinates for Tier 1 is computed. As each (p_i^1, t_i^1, z_i^1)-coordinate is explored, additional rectangular, high-ambiguity regions are identified, which in turn are represented by their (p_j^2, t_j^2, z_j^2)-coordinates (Tier 2). The shortest path that connects these Tier-2 coordinates are inserted within the shorted path for the Tier 1 coordinates as shown in Figure 8d. This process is repeated one more time (Tier 3). For the sake of brevity, the illustration for Tier 3 is skipped in Figure 8.

For an environment with pan, tilt dimensions $m_p \times m_t$, and a $threshold \in \mathbb{R}$, algorithm ***ThreshTargUncert***, shown in Algorithm 12, returns a collection of binary images that represent regions of sufficiently high target uncertainty that would require exploration before the artifacts in these regions can be disambiguated as target or clutter. This procedure is used by ***LazyTSPPath*** to identify a collection of axis-parallel bounding boxes around a binary image that is computed for each threshold in a predetermined set of thresholds $\{threshold_i\}_{i=1}^k$. If any of these bounding boxes violate the camera HFOV restriction, they are broken up into smaller members that do not violate the HFOV restriction. This is done in an ad hoc manner in our implementation. The height, width, and center of any of these bounding boxes can be effectively represented by a (p, t, z)-coordinate using the scheme described previously. The final step of ***LazyTSPPath*** finds the shortest PTZ route $\mathbb{P}$ using a Lazy-TSP procedure. The procedure ***3-Tier-NestedSearch***, shown in Algorithm 13, presents a reactive PTZ route that is based on a three-tier exploration of an initial environment. Each tier has its own predetermined set of thresholds. In Tier 1, a PTZ route $\overline{\mathbb{P}}$ is computed using the image obtained using the ***QuickScan*** procedure. Additional regions that require further exploration to disambiguate target and clutter are identified as each (p, t, z)-coordinate in $\overline{\mathbb{P}}$ is

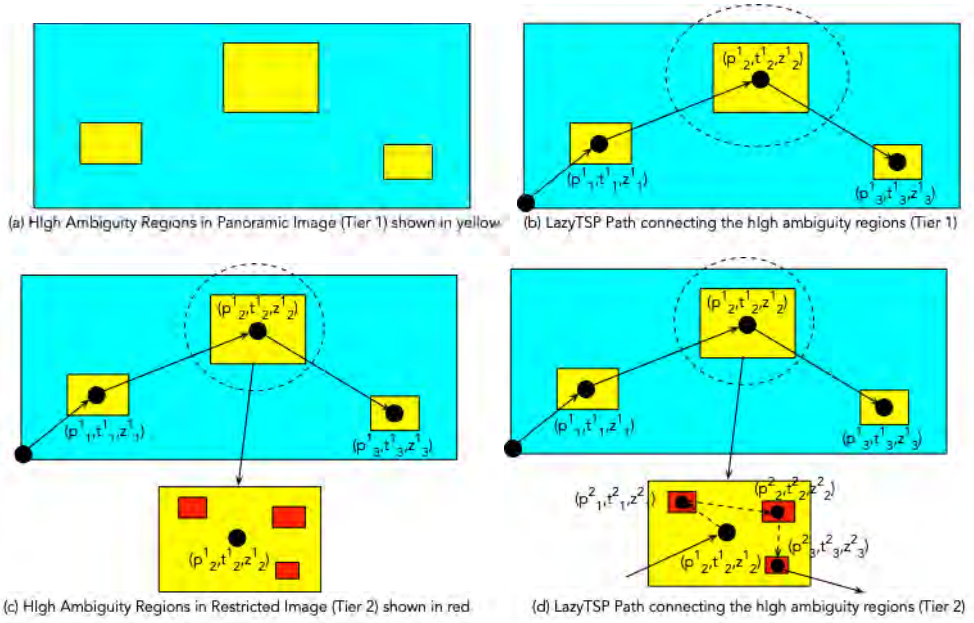


Fig. 8 Illustration of the *LazyTSPPath* algorithm. (a) The rectangular, high-ambiguity regions in the panoramic image are identified for further exploration (Tier 1). (b) The shortest, LazyTSP path that connects these Tier 1 high-ambiguity regions is computed. (c) At each Tier 1 exploration point, additional high-ambiguity regions are identified; this figure illustrates what would happen when the region identified by (p_2^1, t_2^1, z_2^1) is explored (Tier 2). (d) The shortest, LazyTSP path for this region is computed and inserted within the Tier 1 path computed earlier. This process is repeated again (Tier 3), which is skipped in this illustration for brevity.

Algorithm 11 ***LazyTSPPath***$(m_p, m_t,$
$\{threshold_i\}_{i=1}^{k})$
Compute the shortest path between the (p, t, z)*-coordinates identified by the* k*-many thresholds,* $\{threshold_i\}_{i=1}^{k}$*, in an environment with pan, tilt dimensions* $m_p \times m_t$*.*

Initialize the list of path-coordinates $\mathbb{P} \leftarrow NULL$
for $threshold \in \{threshold_i\}_{i=1}^{k}$ **do**
  $\mathbb{I} \leftarrow$ ***ThreshTargUncert***$(m_p, m_t, threshold_i)$
  for each contiguous-image $\widehat{\mathbb{I}} \in \mathbb{I}$ **do**
    Find the bounding box for $\widehat{\mathbb{I}}$ (cf. OpenCV[37]).
    If the bounding box is too large, use a
    covering of smaller bounding boxes that
    meet the camera HFOV in its place.
    Let $\mathcal{L}$ be a list of (p, t, z)-descriptions of these
    bounding boxes.
    $\mathbb{P} \leftarrow \mathbb{P} \cup \mathcal{L}$.
  end for
end for
Reorder $\mathbb{P}$ to reflect the shortest path (using a Lazy-TSP procedure).
return $\mathbb{P}$

explored in the Tier 2. We have chosen to extend this search for an additional tier where a similar process identifies Tier 3 (p, t, z)-coordinates that need further exploration. The resulting path $\mathbb{P}$ is presented as output for evaluation. Other variants include terminating the search in either Tier 1 or Tier 2. The results of our experiments are described in the next section.

4. Experiments

All of our simulations and visual search algorithms have been implemented in Python, except for library calls to the VTK[29] in the case of the 3D simulation, and to the Gurobi Optimizer[36] in the case of Algorithm ***LazyTSPPath***. In this section, we describe the parameters of our experiments and compare the performance of the proposed algorithms.

Algorithm 12 ***ThreshTargUncert*** $(m_p, m_t,$ $threshold)$

Threshold Binary Image of the target uncertainty in an environment with pan, tilt dimensions $m_p \times m_t$.

$\mathbb{M} \leftarrow$ ***QuickScan*** (m_p, m_t) # initialize $\mathbb{M}$
$\mathbb{I} \leftarrow \{0\}^{(m_p \times m_t)}$ # initialize $\mathbb{I}$ to zero-matrix
for $(p, t) \in (m_p \times m_t)$ **do**
  if $(t_{low} < T[p,t] < t_{high})$ **and**
    $(1/(T[p,t] - 0.5)^2 > threshold)$ **then**
    $\mathbb{I}[p,t] = 1$
  end if
end for
return $\mathbb{I} \in \{\mathtt{0}, \mathtt{1}\}^{\mathrm{P} \times \mathrm{T}}$

Algorithm 13 ***3-Tier-NestedSearch*** (m_p, m_t)

Minimal-cost, Context-Aware, Visual Search based on a nested Lazy-TSP procedure for an environment with pan, tilt dimensions $m_p \times m_t$.

Pick a set of thresholds $\{\{threshold_i^j\}_{i=1}^{k_j}\}_{j=1}^{3}$
Initialize path $\mathbb{P} \leftarrow NULL$
Path $\overline{\mathbb{P}} \leftarrow$ ***LazyTSPPath*** $(m_p, m_t, \{threshold_i^1\}_{i=1}^{k_1})$
for $(p, t, z) \in \overline{\mathbb{P}}$ **do**
  $\mathbb{P} \leftarrow \mathbb{P} \circ (p, t, z)$ # concatenate $\mathbb{P}$ and (p, t, z)
  $\widehat{\mathbb{P}} \leftarrow$ ***LazyTSPPath*** $(m_p, m_t, \{threshold_i^2\}_{i=1}^{k_2})$
  for $(\widehat{p}, \widehat{t}, \widehat{z}) \in \widehat{\mathbb{P}}$ **do**
    $\mathbb{P} \leftarrow \mathbb{P} \circ (\widehat{p}, \widehat{t}, \widehat{z})$ # concatenate $\mathbb{P}$ and $(\widehat{p}, \widehat{t}, \widehat{z})$
    $\widetilde{\mathbb{P}} \leftarrow$ ***LazyTSPPath*** $(m_{\widehat{p}}, m_{\widehat{t}}, \{threshold_i^3\}_{i=1}^{k_3})$
    $\mathbb{P} \leftarrow \mathbb{P} \circ \widetilde{\mathbb{P}}$ # concatenate $\mathbb{P}$ and $\widetilde{\mathbb{P}}$
  end for
end for
return $\mathbb{P}$

4.1 Simulation Parameters

The 3D simulation is used to create a performance model of the YOLO object detector.[2] The parameters of the 3D simulation that are most relevant to this task are the following. The diameter of the environment is 1 km, which is identically the maximum distance between a camera and any object in the environment. The number of distinct photo-realistic textures used in the 3D simulation are 25 ground surfaces, 4 roads types, 38 buildings, 35 plants, 87 people, 33 animals, 7 airborne objects, 54 miscellaneous objects (clutter), 8 fences and barriers, 10 street signs, 5 distant backgrounds, and 6 sky textures. The actual number of objects appearing in any environment is random, but the number of people (the targets) is expected to be 20% of the total number of objects excluding buildings and plants. The time of day is varied from 6 am to 6 pm, generating a variety of lighting conditions and shadows. The PTZ camera has a resolution of 1920×1080 pixels, and a horizontal FOV ranging from 2.3° to 64° ($\zeta_{\min}, \zeta_{\max}$), as is common in some commercial zoom cameras. With these parameters, there are occasions when targets in view of the camera occupy only a few image pixels and are nearly impossible to detect when the camera is set to a wide FOV, but are easily detected when the camera is zoomed in.

For the 2D simulation, the field of regard is $180^\circ \times 35^\circ$ (horizontal $\times$ vertical), and this is quantized into 1800×350 ($N_P \times N_T$) cells of size $0.1^\circ \times 0.1^\circ$ ($\Delta s = 0.1^\circ$). The depth of objects from the camera ranges from 2 m to 1 km, with clutter being twice as frequent as targets. The resolution of the camera in the 2D simulation replicates that of the 3D simulation, with an HFOV ranging from 23 to 64 cells.

4.2 Genetic Algorithm Parameters

The GAs have a few parameters that affect their performance. These include the population size, n_{pop}, the number of generations, n_{gen}, the relative mutation rates, m_p, and the fitness function scale factors, α_{d} and α_{s}. From Fig. 9, we see that 200 generations of ***GASearch*** is sufficient to produce very good coverage ($\Sigma(\hat{\boldsymbol{Z}} \leq \bar{\boldsymbol{Z}})$ in Eq. 7) with additional generations mostly reducing the distance traveled ($\Sigma \| \boldsymbol{p}_i^j - \boldsymbol{p}_i^{j-1} \|$) and number of images processed. The remainder of the GA parameters were calibrated empirically. The parameters used for the experiments were $n_{\text{pop}} = 200$, $n_{\text{gen}} = 200$, $m_p = (0.067, 0.067, 0.2, 0.067, 0.067, 0.067, 0.067, 0.2, 0.2)$, $\alpha_{\text{d}} = 2.0$, and $\alpha_{\text{s}} = 0.02$.

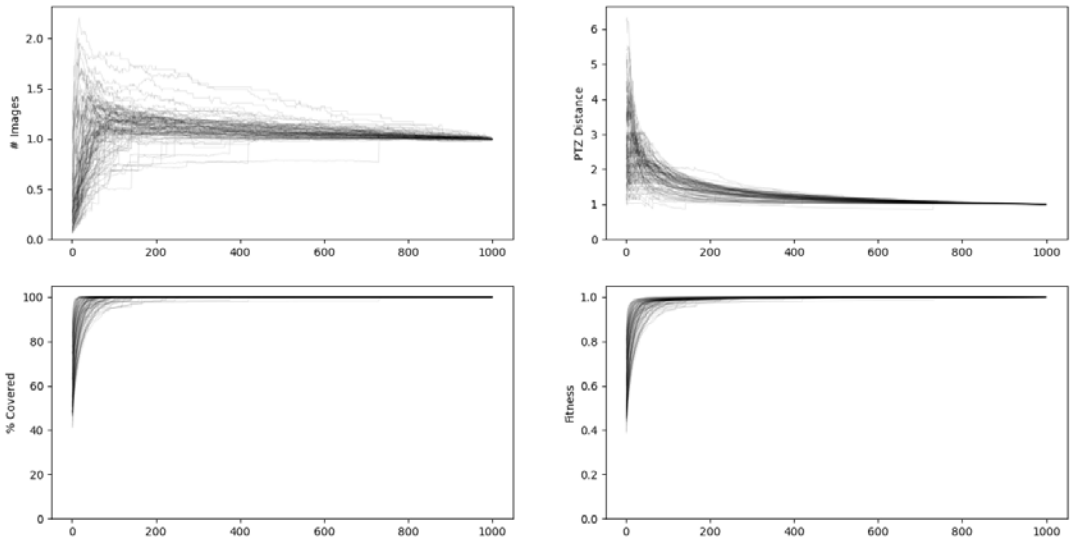


Fig. 9 Performance of best *GASearch* solution in 100 random environments as a function of generation number (along horizontal axis). The top row shows the number of images (left) and distance traveled in PTZ space (right). The bottom row shows the percent of covered cells (left) and solution fitness (right). Data points in all plots except `% Covered` are scaled relative to the final solution at generation 1000.

These parameters are used for both the static and dynamic versions of the GA.

4.3 LazyTSP Path Algorithm Parameters

The camera's HFOV restricts the width of the bounding box to be no greater than 640 pixels. The largest height of the bounding box will be $640 \times (1080/1920) = 360$ pixels. If the bounding box computed in ***LazyTSPPath*** exceeds these dimensions, we replace this single bounding box by a covering of bounding boxes whose width (resp. height) is no larger than 320 pixels (resp. 180 pixels). There are other ways of overcoming the HFOV restrictions, but we opted to use this in our simulations.

As mentioned previously, there are three-tiers to ***LazyTSPPath***, and each tier has a

set of thresholds that are used to identify the bounding boxes, which are then used to identify a (p, t, z)-path. We use the same set of thresholds for each tier, that is $\{threshold_i\}_{i=1}^{3} = \{10, 50, 100, 250, 500, 750\}$. The first-tier search identifies an optimal (p, t, z)-path. During the course of exploration of this path, other regions of interest are selected for further exploration in the second tier. The resulting optimal, second-tier path is intercalated between the appropriate pair of first-tier (p, t, z)-points. This process is repeated another time, where the first-to-second-tier operations are repeated for the second-to-third-tier operations as well.

4.4 Performance Comparison

We ran each of the four algorithms on 100 identical simulated urban environments. Some of these environments are shown in Fig. 10. In all, this experiment included 5275 clutter objects and 2763 target objects. For performance metrics, we use target and clutter recall (detection) rates: the percent of targets that are positively identified as targets, and the percent of clutter that are positively identified as clutter. We set $t_{\text{conf}} = 0.9$, so true targets are considered "positively identified" if their estimated target probability (Eq. 5) is 0.9 or higher, and true clutter is considered positively identified as clutter (i.e., "suppressed") if their estimated target probability is 0.1 or lower. The time taken to perform a visual search, also called PTZ distance, is compared as it is important for many applications. We do not compare the actual run-times of the algorithms since some are implemented fully in interpreted Python while others use compiled and optimized libraries for the computationally expensive parts of the algorithm.

Figure 11 compares the performance of the four visual search algorithms on each of the experiment's 100 trials, and Table 1 summarizes their performance.* We see that ***StaticGA*** produces the shortest search paths (fastest search times), but has the worst target and clutter detection rates. ***QuadtreeDecomp*** has the best overall target and clutter detection rates, but its search paths are, on average, twice as long as those of ***StaticGA***. ***DynamicGA***, however, with only 40% longer search paths than ***StaticGA***, produces target and clutter detection rates that are nearly as good as ***QuadtreeDecomp***. Finally, ***LazyTSPPath*** produces target and clutter detection rates that lie between those of ***StaticGA*** and ***DynamicGA***, but with search paths

*Due to the close visual similarity between some clutter and target objects (e.g., real people and statues of people), the object detector is unable to correctly classify all objects (see detector performance in Figure 5), so perfect detector performance is not possible, even when provided with very high resolution images from highly zoomed PTZ cameras.

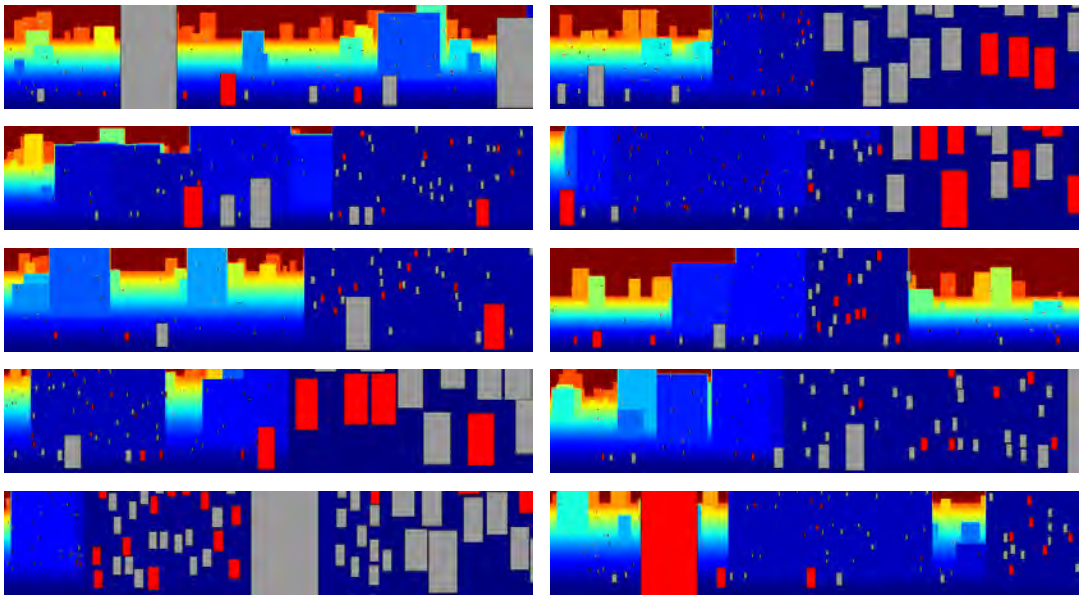


Fig. 10 Depth maps of some simulated urban environments with targets (red rectangles) and clutter (gray rectangles) overlayed. Each spans a $180°$ horizontal view by $35°$ vertical view.

Algorithm	D_r	R_t	R_c
QuadtreeDecomp	200.4	98.11	99.45
StaticGA	100.1	82.43	81.66
DynamicGA	140.1	98.14	98.14
LazyTSPPath	341.0	94.96	90.81

Table 1 Mean performance of the visual search algorithms over 100 trails. D_r is the mean of the solution distance relative to the distance of the shortest solution from among the four algorithms (lower is better). R_t is the mean target recall rate. R_c is the mean clutter recall rate (percent of clutter objects that are suppressed; higher is better).

that are more than twice as long as ***DynamicGA***'s. The poor detection rates of ***StaticGA*** relative to the other three algorithms is due to ***StaticGA*** choosing its visual search path based on a static environment model, which depends only on the initial wide-angle scan of the environment; it does not adapt its search plan to new information obtained during the execution of that search. All of the other algorithms do adapt, dynamically during their search, to information obtained in new views of the environment.

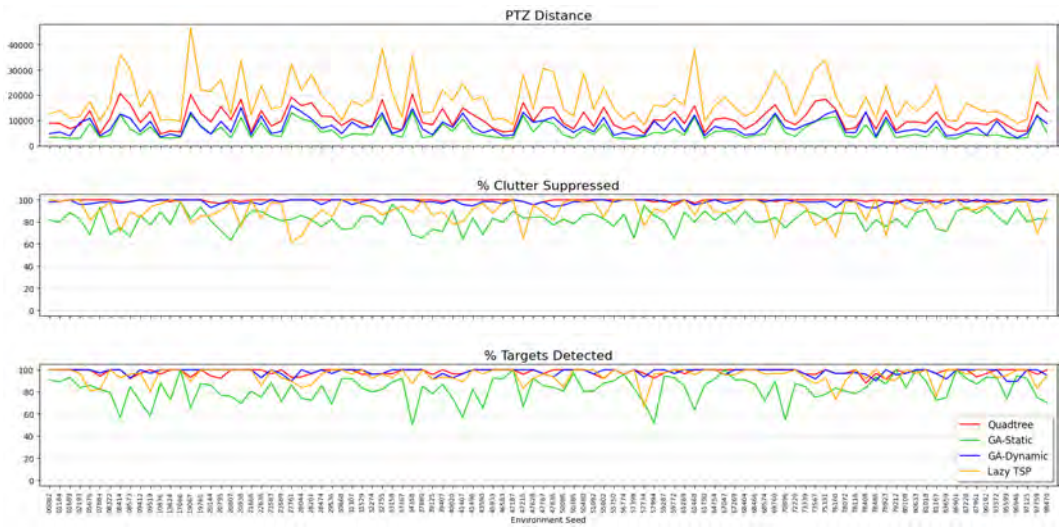
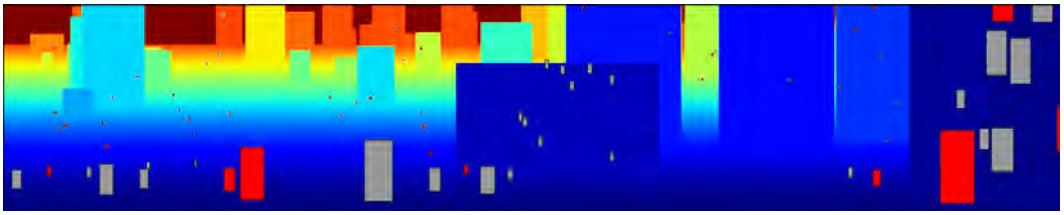


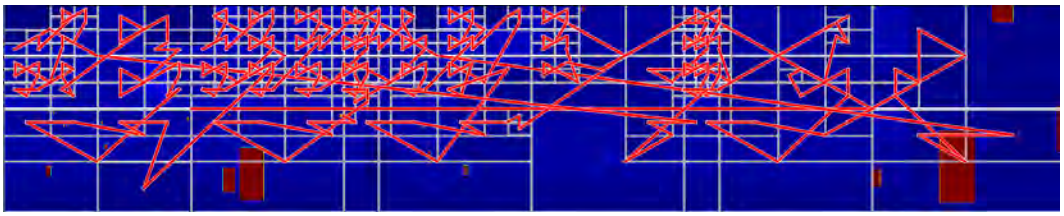
Fig. 11 Performance of the visual search algorithms on 100 random environments

Figure 12 shows the PTZ routes selected by each algorithm on one representative environment. A high-performing algorithm will, after completing its visual search, have assigned very low probabilities (dark blue regions in this figure) to non-target locations and very high probabilities (red regions in this figure) to target locations. Also, high-performing algorithms will use a minimum number of images and minimize the distance traveled in PTZ space needed to acquire those images. Compared to the GA algorithms, the search paths of ***QuadtreeDecomp*** are generally longer due to the recursive NE-SE-NW-SW order in which ***QuadtreeDecomp*** explores quadrants of ambiguous views. Also, ***QuadtreeDecomp*** may require more views than the GAs because the GAs have greater flexibility in the placement of views. The two GAs display similar search paths, but it can be seen that ***StaticGA*** leaves a few targets undetected and a few clutter unsuppressed. The path of ***LazyTSPPath*** seems somewhat erratic in that it views some parts of the scene repeatedly while avoiding other parts, and some high-zoom views are placed needlessly in foreground areas of low depth; it does, however, do a good job detecting targets and suppressing clutter.

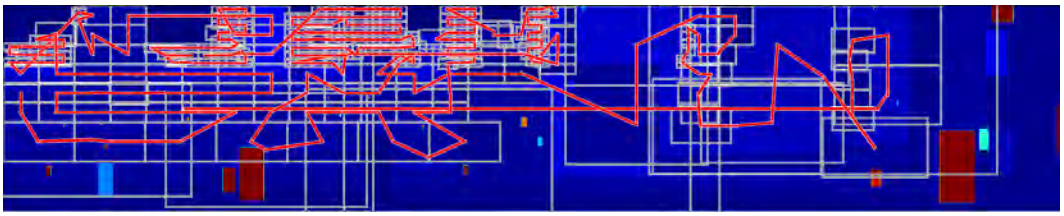
Figure 13 plots, for each algorithm and each of the 100 trials, the target detection rate and clutter suppression rate as a function of the PTZ distance traveled. A good performing algorithm will have these points clustered in the upper left corner of the two plots. ***QuadtreeDecomp*** and ***DynamicGA*** both do well in this regard. Figure 14 shows the final assigned target probabilities for all of the 5275 clutter objects and



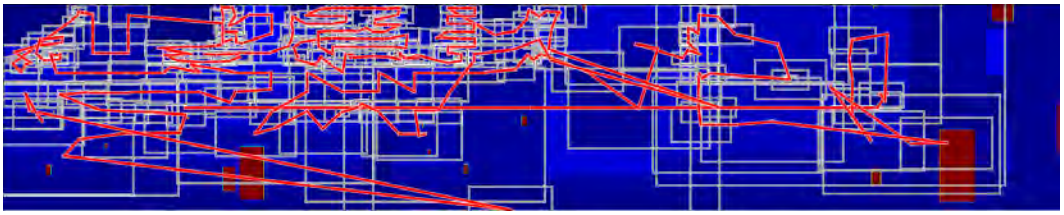
(a) Panoramic of depths with clutter (gray) and targets (red) overlayed



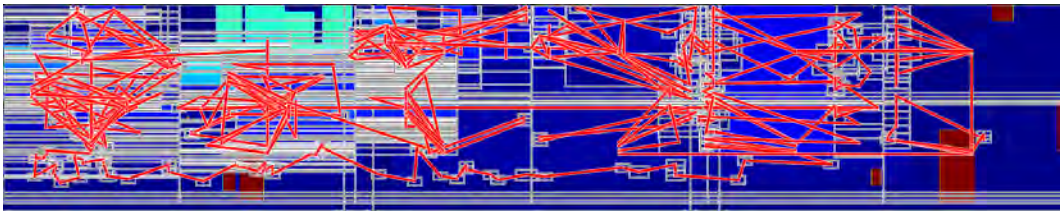
(b) ***QuadtreeDecomp****: 291 images, 18177 path length, 97.7% clutter suppressed, 96.4% targets detected.*



(c) ***StaticicGA****: 279 images, 11661 path length, 77.3% clutter suppressed, 82.1% targets detected.*



(d) ***DynamicGA****: 281 images, 12860 path length, 97.7% clutter suppressed, 96.4% targets detected.*



(e) ***LazyTSPPath****: 647 images, 38373 path length, 86.4% clutter suppressed, 96.4% targets detected.*

Fig. 12 Example solutions on one random environment (#32755). (a) Panoramic image of depths with clutter and target positions overlayed. The depth panoramic is the main input to the PTZ visual search algorithms; the target and clutter locations are hidden from the algorithms. (b)–(e) Final target probability panoramics for each of the four algorithms. PTZ camera views are shown with white rectangles and the path of the camera in pan-tilt space is represented with a red line connecting the centers of these rectangles.

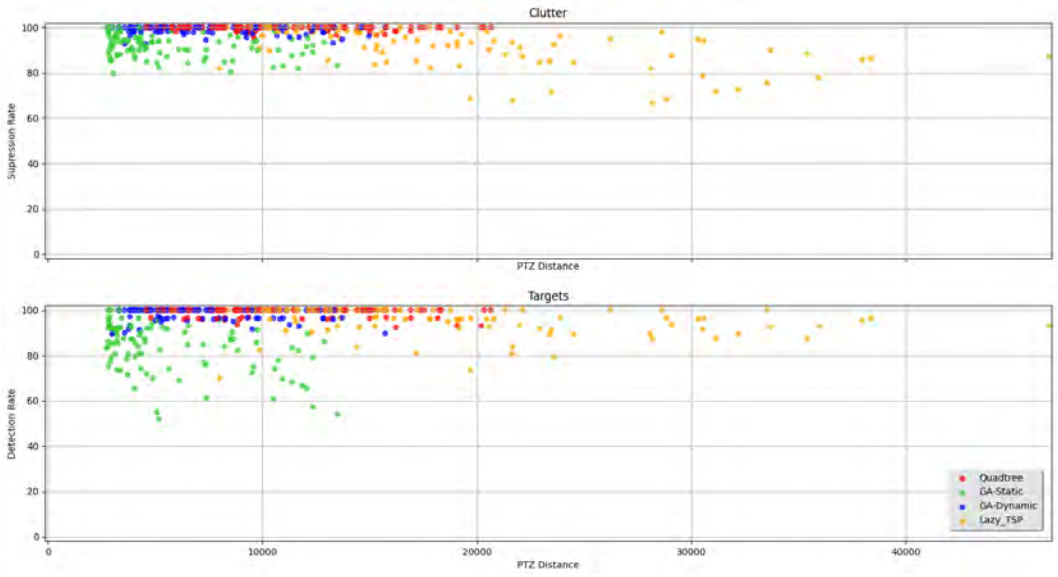


Fig. 13 Performance as a function of PTZ distance: clutter suppression rate (top) and target detection rate (bottom) for each algorithm applied in each of the 100 random urban environments

2763 target objects as a function of depth of the objects from the camera. All of the algorithms except ***LazyTSPPath*** exhibit a ramping effect where, at various depths, there are large drops in the target probability assigned to clutter objects. This is due to the changes in camera zoom at those depths. This is most prominent with ***QuadtreeDecomp*** as it has the least flexibility in selecting zoom values, always halving the previous zoom when it needs to zoom-in in ambiguous regions of the environment.

5. Conclusion and Future Work

In this report we have formalized the PTZ visual search problem, a problem that has informally existed for many years, but has recently seen a need to be fully automated as the number of PTZ surveillance cameras has significantly increased. We developed a compact simulation environment that enables easy large-scale benchmarking of visual search algorithms under close to real-world conditions. We then proposed four algorithms for PTZ visual search: a quadtree decomposition algorithm, two varieties of a genetic algorithm, and a TSP algorithm using lazy constraints. These algorithms were evaluated on a set of 100 random outdoor environments. The results show that, among the four algorithms, ***DynamicGA*** is the most efficient in terms of detection of targets and suppression of clutter relative

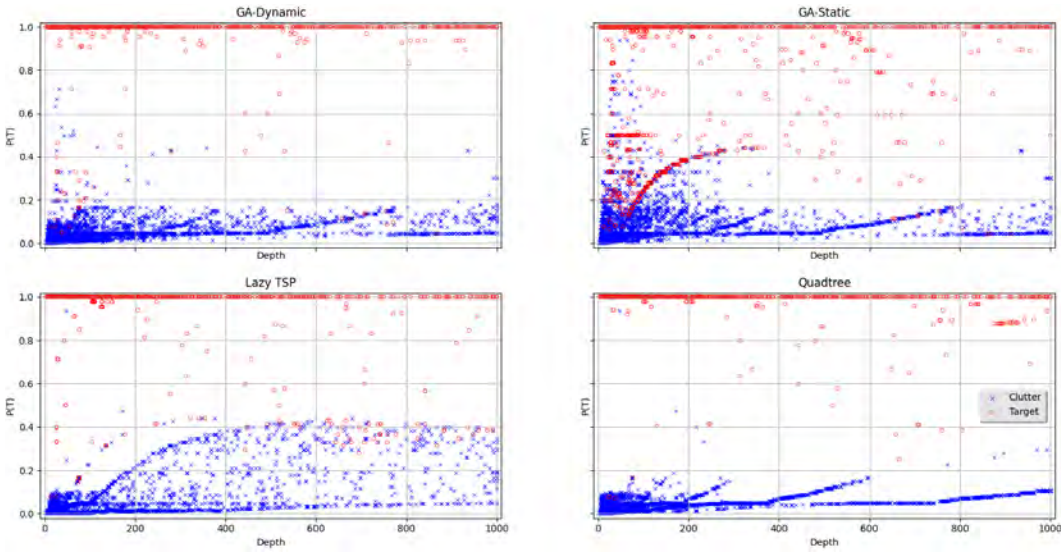


Fig. 14 Final assigned target probabilities for all true clutter objects (blue "x"s) and true target objects (red "o"s) as a function of depth of the objects from the camera. Ideally, target objects should have assigned target probabilities close to 1.0 and clutter objects should have assigned target probabilities close to 0.0.

to the time needed to acquire and process those images. However, ***DynamicGA*** has the disadvantage that its PTZ route planning requires more central processing unit (CPU) resources than some of the other algorithms.

For future work, we would like to test our algorithms in real-world environments and extend these algorithms to visual search of both static and moving objects from moving platforms, including mobile robots. We are also interested in using the depth and semantic label images to implement additional constraints and priorities in the search process, such as to allow different priorities for various events in specific parts of the environment. For example, a user may want to prioritize quickly locating people on the rooftops of distant buildings while still searching the entire environment for lower priority events.

6. References

1. Mukhina KD, Visheratin AA, Nasonov D. Orienteering problem with functional profits for multi-source dynamic path construction. PLOS ONE. 2019;14(4):1–15.

2. Bochkovskiy A, Wang CY, Liao HYM. Yolov4: optimal speed and accuracy of object detection. ArXiv. 2020;abs/2004.10934.

3. Hu J, Hu S, Sun Z. A real time dual-camera surveillance system based on tracking-learning-detection algorithm. In: 25th Chinese Control and Decision Conference (CCDC); 2013; p. 886–891.

4. Hu S, Shimasaki K, Jiang M, Senoo T, Ishii I. A simultaneous multi-object zooming system using an ultrafast pan-tilt camera. IEEE Sensors Journal. 2021;21(7):9436–9448.

5. Scotti G, Marcenaro L, Coelho C, Selvaggi F, Regazzoni C. Dual camera intelligent sensor for high definition 360 degrees surveillance. IEE Proceedings - Vision, Image and Signal Processing. 2005;152:250–257(7).

6. Tarhan M, Altug EA. Catadioptric and pan-tilt-zoom camera pair object tracking system for UAVs. Journal of Intelligent and Robotic Systems. 2011;61:119–134.

7. Bueno MB, Giró-i Nieto X, Marqués F, Torres J. Hierarchical object detection with deep reinforcement learning. Deep Learning for Image Processing Applications. 2017;31(164):3.

8. Wu Z, Khan NM, Gao L, Guan L. Deep reinforcement learning with parameterized action space for object detection. In: 2018 IEEE International Symposium on Multimedia (ISM); p. 101–104.

9. Caicedo JC, Lazebnik S. Active object localization with deep reinforcement learning. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); p. 2488–2496.

10. Wu ZZ, Wang XF, Zou L, Xu LX, Li XL, Weise T. Hierarchical object detection for very high-resolution satellite images. Applied Soft Computing. 2021;113:107885.

11. Sommerlade E, Reid I. Information-theoretic active scene exploration. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition; p. 1–7.

12. Salvagnini P, Pernici F, Cristani M, Lisanti G, Del Bimbo A, Murino V. Non-myopic information theoretic sensor management of a single pan-tilt-zoom camera for multiple object detection and tracking. Computer Vision and Image Understanding. 2015;134:74–88; Image Understanding for Real-world Distributed Video Networks.

13. Golden B, Levy L, Vohra R. The orienteering problem. Naval Research Logistics. 1987;34:307–318.

14. Ciancio C, Maio AD, Laganà D, Santoro F, Violi A. A Genetic algorithm framework for the orienteering problem with time windows. In: New trends in emerging complex real life problems; Springer; 2018; p. 179–188.

15. Gunawan A, Lau HC, Vansteenwegen P. Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research. 2016;255(2):315–332.

16. Kobeaga G, Merino M, Lozano JA. An efficient evolutionary algorithm for the orienteering problem. Computers & Operations Research. 2018;90:42–59.

17. Tasgetiren M, Smith A. A genetic algorithm for the orienteering problem. In: Proceedings of the 2000 Congress on Evolutionary Computation (CEC00); Vol. 2; p. 910–915.

18. Judd T, Ehinger K, Durand F, Torralba A. Learning to predict where humans look. In: 2009 IEEE 12th International Conference on Computer Vision; p. 2106–2113.

19. Ranzato M. On learning where to look. ArXiv. 2014;abs/1405.5488.

20. Mousavi S, Schukat M, Howley E, Borji A, Mozayani N. Learning to predict where to look in interactive environments using deep recurrent q-learning. ArXiv. 2016;abs/1612.05753.

21. Qaiser T, Rajpoot NM. Learning where to see: A novel attention model for automated immunohistochemical scoring. IEEE Transactions on Medical Imaging. 2019;38:2620–2631.

22. Uzkent B, Ermon S. Learning when and where to zoom with deep reinforcement learning. In: Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition; p. 12345–12354.

23. Capel D. Image mosaicing. In: Image mosaicing and super-resolution; Springer London; 2004. p. 47–79.

24. Dong X, Garratt MA, Anavatti SG, Abbass HA. Towards real-time monocular depth estimation for robotics: A survey. IEEE Transactions on Intelligent Transportation Systems. 2022;23(10):16940–16961.

25. Ren S, He K, Girshick R, Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R, editors. Advances in Neural Information Processing Systems; Vol. 28; Curran Associates, Inc.; 2015.

26. Cetinkaya B, Kalkan S, Akbas E. Does depth estimation help object detection?. Image and Vision Computing. 2022;122:104427.

27. Tong K, Wu Y. Deep learning-based detection from the perspective of small or tiny objects: A survey. Image and Vision Computing. 2022;123:104471.

28. Wallach H. Evaluation metrics for hard classifiers. Cambridge: Cavendish Laboratory, University of Cambridge. 2006;.

29. Schroeder W, Martin K, Lorensen B. The visualization toolkit, 4th edn. kitware. New York. 2006;.

30. Holland JH. Genetic algorithms. Scientific American. 1992;267(1):66–73.

31. Xu Z, Liao Q. Gaussian process based expected information gain computation for Bayesian optimal design. Entropy. 2020;22(2):258.

32. Leung Y, Gao Y, Xu ZB. Degree of population diversity - a perspective on premature convergence in genetic algorithms and its markov chain analysis. IEEE Transactions on Neural Networks. 1997;8(5):1165–1176.

33. Malik S, Wadhwa S. Preventing premature convergence in genetic algorithm using DGCA and elitist technique. International Journal of Advanced Research in Computer Science and Software Engineering. 2014;4(6).

34. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. MIT press; 2022.

35. Pferschy U, Staněk R. Generating subtour elimination constraints for the tsp from pure integer solutions. Central European Journal of Operations Research. 2017;25(1):231–260.

36. Glockner G. Lazy constraint generation using the Gurobi optimizer. https://support.gurobi.com/hc/en-us/articles/360013197972-How-do-I-implement-lazy-constraints-in-Gurobi-, Accessed on 2022-09-08.

37. OpenCV. Creating bounding boxes and circles for contours. https://docs.opencv.org/3.4/da/d0c/tutorial_bounding_rects_circles.html, Accessed on 2022-09-08.

List of Symbols, Abbreviations, and Acronyms

2D	two-dimensional
3D	three-dimensional
BILP	Binary Integer Linear Program
CPU	central processing unit
EDGCA	Elitist and Dynamic Genetic Clustering Algorithm
GA	genetic algorithm
HFOV	horizontal field of view
IoT	Internet-of-Things
IoU	Intersection over Union
NP	Nondeterministic Polynomial-time
OP	orienteering problem
OPFP	Orienteering Problem with Functional Profits
ppm	pixels per meter
PTX	pan-tilt-zoom
PTZSP	PTZ Search Problem
ROI	region of interest
TSP	traveling salesman problem
VFOV	vertical field of view
VTK	Visualization Toolkit
YOLO	you only look once

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD-RLB-CI TECH LIB
2 (PDF)	DEVCOM ARL FCDD-RLA-JB P DAVID D BARAN
1 (PDF)	DEVCOM ARL FCDD-RLA-IB A HARRISON