Target Localization with Drones using Mobile CNNs

Yongxi Lu[1], Zeyangyi Wang[1], Ziyao Tang[1] and Tara Javidi[1]

***Abstract*— Fast and accurate visual search is an enabler for many applications of drones. Prior works use POMDPs to produce effective search strategies. As the observation models are from heuristics, the robustness of these approaches on the field is unclear. This work builds a testbed that combines latest developments in related areas, including mobile CNNs for inference on mobile platforms and policy search with point based methods, in a POMDP framework. A dataset for a simple but realistic application, search for a single basketball, is collected to train the perception modules, investigate their error characteristics and validate the control algorithm. Simulation using realistic parameters from data sheds lights on the importance of considering persistent factors in the environment in modeling the observations from the CNN-based perception module, which is not systematically considered previously.**

I. INTRODUCTION

In recent years commercial quadcopters, or drones, have become widely available. These flying machines perceive the world through a unique perspective and unlike ground vehicles, are not subject to the usual traffic patterns. This has motivated a wide range of applications using drones. Recently, drones have been applied to search and rescue [1]–[4], active classifications [5], [6], aerial surveillance [7], [8] and agriculture [9], to name few. An enabler of many such applications is an algorithm that allows the drone to recognize and approach a pre-defined target quickly. Similar problems have been investigated in prior works [10]–[15]. Closest to our setting is [14], [15]. However, Sudevan *et al*. [14] uses a classical perception module, which is less effective for complex applications. Gupta *et al*. [15] assumes a simple observation model without validation on the physical environment. In view of these limitations of prior works, this paper makes the following contributions:

- A dual-mode image processing mechanism is fully integrated into our Parrto Bebop 2 drone and the accompanying Samsung S8 phone. The first mode is faster but with less accuracy, while the second mode is more expensive but significantly more accurate. This is in line with prior work in detection and tracking on drones [4], [16], [17] which performs target search through image processing. Here our contribution is to extend the prior work to scenarios in which the search area is large, making it necessary to collect images at different flight locations.
- An image classifier trained on images collected from actually operating our drone. The classifier is built on state-of-the-art computer vision algorithms, a MobileNet CNN [18]. From the data collected, we note that the performance of the trained classifier is a function of flight altitude as well as random but persistent environmental factors that affects visibility. The former is in line with the assumptions in prior works, but we observe interesting differences. The latter is a novel aspect that has not been considered before.
- A partially observable Markov decision problem formulation. The proposed formulation incorporates the acquisition process as well as the classifiers' characteristics. Our work is in line with a large body of research on motion planning for drones using POMDPs [10]–[13], [15], [19], but our focus is on data collection and analysis that leads to a realistic model. Another novelty is the adoption of latest computer vision algorithms.
- A thorough comparative analysis of the impacts of the noise and the robustness to modeling artifacts. Through elaborate simulations, we investigate the impact of the persistent environmental factors in search using drones.

The paper is organized as follows: Section II presents the problem formulation, highlights its difference to prior works and introduces our testbed. Section III discusses our simulation which shows the importance of considering the persistent factors in the environment, such as visibility. Section IV introduces our dataset collection process and the perception module design. Section V concludes the paper and discusses interesting future directions. The attached demo video shows a successful test flight of our system.

II. TARGET SEARCH PROBLEM

We consider the problem of optimization the flight path of a drone which is tasked with localizing a single static target on the ground in the face of uncertainty and physical constraints of the system. The components of our model include

- **Search Area** We consider the location of a single target of interest in a two-dimensional plane on the ground, denoted as $I_{\text{target}} = [I^{\text{la}}_{\min}, I^{\text{la}}_{\max}] \times [I^{\text{lo}}_{\min}, I^{\text{lo}}_{\max}]$. The static target is denoted as $Y_{\text{target}} \in I_{\text{target}}$.
- **Flight Space** We consider a three-dimensional region $I_{\text{flight}} = [I^{\text{la}}_{\min}, I^{\text{la}}_{\max}] \times [I^{\text{lo}}_{\min}, I^{\text{lo}}_{\max}] \times [I^{\text{alt}}_{\min}, I^{\text{alt}}_{\max}]$. This denotes the allowable space of flight for the drone. The location of the drone at time t is denoted as $X_t \in I_{\text{flight}}$.
- **Actions** At every time step, the drone is allowed to move to one of the immediate neighbors of its current location, denoted as $N(x_t)$. It will also choose a sensing mode $a_{\text{sense},t}$ from a finite set of sensing modes. The sensing modes differs in their cost and reliability. The

[1] The authors are with the Department of Electrical and Computer Engineering, University of California, San Diego. {yol070, zew026, zit021, tjavidi}@eng.ucsd.edu

drone can also choose to stop its operation and land at any time.

- **Observations** At any given time t and flight location $X_t \in I_{\text{flight}}$, the drone can acquire and process and image to arrive at a binary observation $O_t \in \{0, 1\}$. This observation is an indication of whether the target is visible in the field-of-view (FOV) of the drone at this time step. We denote $Y_t \in \{0, 1\}$ be the random variable denoting the "true" inclusion indicator, which depends on X_t and Y_{target}. O_t is a noisy version of Y_t.
- **Observation Noise** We take a probabilistic model to describe O_t as a function of the altitude of the drone (denoted as H_t), the visibility V that reflects environmental conditions, and the true inclusion indicator Y_t:

$$\begin{aligned}&\mathbb{P}(O_t = 1 | Y_t = y, H_t = h, V = v) \\ &= \begin{cases} 0 & \text{if } h \geq v \\ p_{0\to1}(h) & \text{if } h < v, y = 0 \\ 1 - p_{1\to0}(h) & \text{if } h < v, y = 1 \end{cases}\end{aligned} \tag{1}$$

  We emphasize that $p_{0\to1}(h)$ and $p_{0\to1}(h)$ is more precisely a function of the sensing mode selected. We omit this in the notation for simplicity.
- **Prior distribution** We assume the target is uniformly distributed in I_{target}. We also assign a Bayesian prior to V, denoted as p_V. In practice this prior shall be estimated from data. For multiple sensing modes, a prior is assigned for each independently.
- **Reward** The drone receives a step-wise movement cost and sensing cost. The former is dependent on the speed of the drone. The latter is dependent on the sensing mode design. Both will be discussed in greater details later. If the drone chooses to stop, it receives a positive reward if: (a) $H_t = I^{\text{alt}}_{\min}$ (flying at minimum altitude). (b) $Y_t = 1$ (the current FOV includes the target).

We note that our problem formulation is similar to the one investigated in the recent work [15]. However, our model is significantly different in its observation model. In particular, it has a notion of "visibility" which is not considered in prior works. Our model says that for each flight session, there is an intrinsic maximum visibility level of the current conditions. If the drone is flying on or above this level, the perception module will output a trivial "0" consistently. This is motivated by our observations from field tests and data collection practice (see Section IV for more details). Our formulation models the effects of "persistent factors" in the environment. Those factors could be lighting conditions, background and weather. Presented with these conditions, the perception module can lose its ability to successfully detect the target, either due to reduced image quality or a lack of data points in the training set that represents the current condition. We note that the latter is particularly prevalent in a modern, data-drive perception module, such as the mobile CNN algorithm we adopted. While such algorithms are much more successful in complex tasks compared to classical algorithms, they could be more susceptible to the kind of catastrophic failures we consider in this work due to their internal complexity and requirement for large amount of data. In contrast, prior work assumes the observation noise is conditionally independent given the height and the target location. It does not consider persistent factors in the observation model. Their simplified model is unrealistic. From our simulations to be discussed in Section III, ignoring those factors could lead to sub-optimal search algorithms.

A. Visibility-Aware POMDP

Our formulation can be easily casted into a POMDP, more precisely a mixed observability Markov decision processes (MOMDPs) [20]. The problem is specified by a tuple $\mathbf{P} = (S, \mathbb{P}_0, A, T, \Omega, H, R, \gamma)$, denoting states, initial state distributions, actions, transition function, set of observations, observation function, reward function and discount factor, respectively.

- **States:** The states $S = (\mathcal{X} \cup \{\xi, \Xi\}) \times \mathcal{Y} \times V$. We use quantization to simplify the problem domain. In particular, $\mathcal{X} \subseteq I_{\text{flight}}$, $\mathcal{Y} \subseteq I_{\text{target}}$ are discrete grid points within the flight space and the target space, respectively. We use a $N_{\text{la}} \times N_{\text{lo}} \times N_{\text{alt}}$ grid for $\mathcal{X}$, and its $N_{\text{la}} \times N_{\text{lo}}$ projection onto the ground plane for $\mathcal{Y}$. V is the space of visibility levels. Since $\mathcal{X}$ has N_{alt} distinct altitudes, V can be described by $(N_{\text{alt}} + 1)^{N_{\text{sense}}}$ intervals covering $[0, \infty)$ for each sensing mode (N_{sense} denotes the number of sensing modes). ξ and Ξ is the starting and ending states, respectively. The $\mathcal{X} \cup \{\xi, \Xi\}$ factor is observable, while $\mathcal{Y}$ and V are hidden parts of the state space.
- **Initial Distributions:** The observable state is initialized at ξ, the starting state. The target $Y_{\text{target}} \in \mathcal{Y}$ follows uniform distribution, the visibility has a prior p_V.
- **Actions:** Since $\mathcal{X}$ is a grid point, the neighbor $N(X_t)$ is the four neighbors at the same altitude and the location directly above and below the current location. Thus the movement action can be specified by $\{\text{left}, \text{right}, \text{forward}, \text{backward}, \text{up}, \text{down}$, denoted as A_{fly}. The drone also decides on the sensing modes for the next location, denoted as A_{sense}. The stopping action is denoted as Δ. The action space is thus $A = A_{\text{fly}} \times A_{\text{sense}} \cup \{\Delta\}$.
- **Transition Function:** If $s = \xi$, the drone will move to the highest location at the center of the flight space. If $a = \Delta$ or if a_{fly} leads to a location outside $\mathcal{X}$ the next state is Ξ (the end state). The drone will stay at the end state once entered. Otherwise the drone will move to the location specified by a_{fly} (thus changing the observable state). The hidden states are always static.
- **Observations:** The observation set $\Omega = \{0, 1, nil\}$.
- **Observation Function:** The observation is nil if and only if the next state is the stopping state nil. Otherwise the observation distributes according to Eqns. 1.
- **Reward Function:** The reward function is introduced in the problem formulation. We use realistic measurements to acquire movement and sensing costs. The positive reward for successful search is a hyperparameter.

- **Discount** We set the discount factor γ to 0.99.

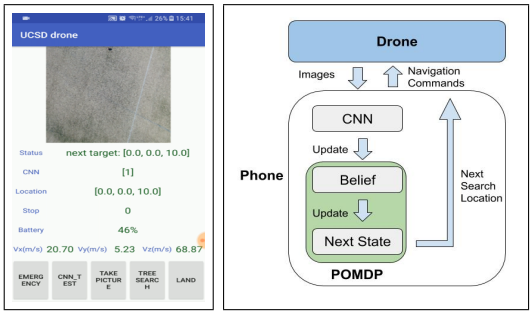

Fig. 1: The control interface. **Left**: Screenshot of the Android app. The app supports both manual control for data collection as well as autonomous search using the POMDP policies. **Right**: Control loop of the drone.

B. Control System

Approximately optimal solutions are found using the SARSOP solver [21]. We use the APPL toolkit [1]. Model searches are performed in 2-hour sessions following the practice in [15]. The solution of the POMDP problem using SARSOP is a piece-wise linear function that approximates the value function. This function is represented as a set of α-vectors with the length of the hidden states. Following the MOMDP formulation [20] each observable state s_{obs} is associated with a set of α-vector denoted as $\Gamma(s_{\text{obs}})$, each in turn is associated with a unique action $a(\alpha)$. Let the belief vector on the hidden states be b_t, the action is selected by

$$a(\alpha) = \arg\max_{\alpha \in \Gamma(s_{\text{obs}})} (\alpha \cdot b_t) \tag{2}$$

Given the current state, the controller first finds the next action using Eqns. 2. As in our model the next observable states is a deterministic function of the current observable state and the action, the observable state can be updated after the action is known. The drone can then move to the next position (or stop and land), and acquire an observation using the selected sensing mode at the new location. The observation is then used to update its internal belief on the hidden states. This process repeats until a timeout is issued by the meta-controller (to avoid excessively long search sessions), or a stop action is chosen. We implement this control loop on an Android device that communicates with the drone and its onboard camera through a wireless connection. All the control modules as well as the perception modules (to be discussed in Section IV) are implemented using TensorFlow [22] for convenient deployment. Figure 1 shows the UI design of the Android app and the illustration of the control loop for target search.

III. Simulations

We build a simulation environment to investigate the impact of the visibility level in our model. In our simulations, we assume the drone can only fly to the center points of a $7 \times 7 \times 7$ grid of the location space $\mathcal{X}$. The intervals defining the latitude, longitude and altitude ranges are $[-14, 14]$, $[-21, 21]$, $[4, 16]$ respectively, in meters. The drone is assumed to move at a speed of 2 m/s along arbitrary directions. The perception module supports up to two sensing modes, with delays of 0.4 seconds and 2.4 seconds per frame respectively. A 1.4-second delay is added to the cost of each sensing modes to model the communication between the drone and the android device. These movement and sensing delay parameters are realistic measurement in our testbed, and will be discussed in greater details in later sections. To highlight the effect of this structured noise, we perform simulations using a wide range of noise parameters. We set those numbers in a way that is qualitatively similar to real measurements (detailed in Section IV). To evaluate a given policy, we collect two statistics:

- **Localization accuracy** measured as the percentage of test sessions in which the drone stops at 4 meters, and its field-of-view includes the target.
- **Average search time** spent on the sessions. The time is measured using the aforementioned parameters.

An important detail is the sampling of the random variables, such as target locations, maximum visibility levels and observations. In our simulations, the target locations are sampled uniformly in the projection of the location grid onto the ground. Our simulator distinguishes two sampling modes

- **Structured persistent noise**: the maximum visibility level V is sampled first according to the supplied prior distribution, and then the observations are sampled according to Eqns. 1 given the realization of V.
- **Independent noise**: the observations has cross-over probabilities that are set to mimic the average case of the corresponding persistent noise model. More precisely, following Eqns. 3.

$$\begin{aligned} \tilde{p}_{0\to1}(h) &= p_{0\to1}(h)(1 - P_V(h)) \\ \tilde{p}_{1\to0}(h) &= (P_V(h) + p_{1\to0}(h)(1 - P_V(h))) \end{aligned} \tag{3}$$

where P_V is the cdf. of V, and the $p_{0\to1}(h)$ and $p_{1\to0}(h)$ are the false positive and false negative rates at height h, same as in Eqns. 1.

Baseline Algorithms Prior works consider random and heuristic policies as baselines. Another interesting baseline is linear search at the minimum height. However, all of these are much slower and is not the focus of this work, thus we do not include them in our comparison. We mainly compare our work against a simplified POMDP formulation that removes V (the visibility level) from our visibility-aware POMDP formulation. We highlight the importance of considering visibility as a persistent factor in the observation model through comparing with this baseline algorithm.

Meta Parameters for Simulations Unless specified otherwise, all simulations are performed with a timeout period of 180 seconds and the statistics are compiled from 10,000 repetitions.

[1] http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl

A. Advantage of Considering Structured Noise

We first consider a scenario in which the environment is filled with structured noise. The designer of the search algorithm may neglect this structure, and use the baseline algorithm instead. This designer still has access to the empirical observation of the error rates, which is the average of different visibility levels (see Eqns. 3). This is compared against a design based on accurate measurement on p_V and the adoption of visibility-aware algorithm.

We build two sets of test cases to identify situations in which the effects of structured persistent noise are largest. Set one concerns one sensing mode. We set the false positive rate to 0.01, and set the intrinsic false negative rate to a monotonically increasing linear function of height. At 4 meters the false negative rate is 0.025. The rate grows at a step of 0.025 per two meters. These numbers are qualitatively similar to real measurements for the less expensive sensing mode, shown in Figure 5. We assume the maximum visibility level is either at 12 meters or above 16 meters, for the sake of simplicity. Thus the prior on the visibility level is completely specified by $p_v(12)$ where $p_v(12) + p_v(> 16) = 1$. We set $p_v(12)$ to $\{0.2, 0.4, 0.6, 0.8\}$ to investigate different strength of the persistent noise. It is assumed that sensing takes 1.8 seconds per frame (the less expensive sensing mode in the simulator). For set two, we add a more expensive but less erroneous sensing mode to the problem (3.8 seconds per frame including communication cost). It is assumed to have the same intrinsic error rates with the sensing mode in set one, but it always has $p_v(> 16) = 1$. This added sensing mode thus serves as an expensive "backup": if the less expensive sensing mode fails above 12 meters, the "clean" but more expensive sensing mode can kick in. Intuitively, this should enable the drone to use a more aggressive strategy in using the less expensive sensing mode.

TABLE I: Improved Search Time using Visibility

The baseline model has access to the empirical erro r rates averaged over the prior on V. Simulations are performed for structured, persistent noise.

Single	**Time (seconds)**		**Success Rate**	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.2$	44.09	46.76	99.82%	99.81%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.4$	45.72	48.35	99.88%	99.78%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.6$	46.04	52.30	99.92%	99.80%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.8$	46.85	46.91	99.93%	99.89%
Dual	**Time (seconds)**		**Success Rate**	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.2$	39.76	42.82	99.88%	99.72%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.4$	40.08	43.53	99.87%	99.79%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.6$	40.07	47.47	99.79%	99.69%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.8$	40.07	67.02	99.76%	99.82%

Table I summarizes the results. In the single sensing mode case, the baseline algorithm results in longer search time except for when $p_V(12) = 0.8$. The gain is largest when $p_V(12) = 0.6$. Intuitively, when persistent visibility is insignificant (small $p_V(12)$), the advantage of using the visibility-aware formulation is expected to be small. Interestingly, when $p_V(12)$ is very large the gain also diminishes. We conjecture that at this particular situation, the two formulations result in similar policies. Intuitively, when the empirical error rates are large at high altitudes, the drone should quickly descend to avoid wasting time. Access to visibility level is not helpful in this scenario as in most realizations the visibility is poor. This suggests considering persistent factors when their effects are significant but moderate is most important for designing successful search strategies. In the dual sensing mode case however, the visibility-aware formulation yields largest gain when $p_V(12) = 0.8$. The knowledge of the visibility structure seems to allow the drone to evaluate the information from the two sensing modes more accurately.

TABLE II: Time at Varying Realization of Visibility

Comparing search time at different realization of the visibility. $p_V(12)$ shows the prior on visibility that controls the noise model available to the policy search algorithm at the design phase. $V > 16$ can be seen as "good" realization of visibility, as the drone can make gain information from high altitudes. $V = 12$ is correspondingly the "bad" visibility realizations.

	$p_V(12) = 0.2$		$p_V(12) = 0.4$	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$V > 16$	42.58	39.99	44.49	39.88
$V = 12$	50.27	73.67	47.61	60.96
	$p_V(12) = 0.6$		$p_V(12) = 0.8$	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$V > 16$	43.68	40.10	43.82	44.06
$V = 12$	47.59	60.33	47.61	47.63

B. Comparing Policies

It is very interesting to understand why and how the baseline algorithm is inferior when the structured noise we consider is present. While it is in general difficult to understand the numerical solution of a complex POMDP, we provide insights through analysis on the sample path realizations taken by individual policies. For simplicity, we focus on the single sensing mode case.

We first look at the average time required for the search to stop at sessions at varying realization of visibility. We note that in our simple simulation case, the maximum visibility level can either be at 12 meters or above 16 meters. Thus there are only two cases, "good" or "bad" visibility. In Table II we compare the two types of policies. In general, policies from visibility-aware POMDP searches slightly longer in cases where the visibility is good (only invisible above 16 meters), but when visibility is poor baseline policies perform much longer search. This suggests that excessive sensing in cases where visibility is poor is the main reason for the longer overall search time of the baseline policies.

Table III shows the average number of acquisitions performed at different heights. Comparing the two types of

policies, the baseline policies spend more time on or above 12 meters, as well as on the lowest height of 4 meters. From 6 to 10 meters the baseline policies spend less time. This suggests that the visibility-aware policies tend to avoid spending too much time on heights subject to structured noise. The knowledge of visibility level could lead to more accurate belief updates, since if the visibility is found to be poor the observations collected from 12 to 16 meters should be discarded. This could be the reason for the shorter search at the lowest height.

But could the visibility-aware POMDP collect information about the maximum visibility level, given that it spends less observations above 12 meters? To answer this question, we examine the belief vectors of the POMDPs after stopping. We use MAP decoding on the belief vectors and compare against ground truth maximum visibility levels. It seems at termination the POMDPs is in many cases aware of the true visibility levels, as in Table IV which shows accuracies are well above random guessing (which has 50% accuracy).

TABLE III: Acquisitions at Varying Heights

The number of acquisitions (observations) made at different heights. This table shows the simulation under the structured noise. It compares the behavior of the two types of policies in this realistic setting.

	$p_V(12) = 0.2$		$p_V(12) = 0.4$	
Height (meters)	*Visibility*	*Baseline*	*Visibility*	*Baseline*
4	3.72	2.89	2.58	3.11
6	1.44	1.35	2.66	1.56
8	1.70	1.78	1.56	1.58
10	2.35	1.38	3.46	2.21
12	1.03	2.00	1.01	1.01
14	1.00	1.01	1.03	1.00
16	1.74	3.17	1.00	3.58
	$p_V(12) = 0.6$		$p_V(12) = 0.8$	
Height (meters)	*Visibility*	*Baseline*	*Visibility*	*Baseline*
4	2.25	2.73	1.82	2.65
6	3.10	2.17	2.71	2.60
8	1.49	1.85	2.32	1.75
10	3.55	2.19	3.72	3.59
12	1.00	1.00	1.00	1.00
14	1.00	1.00	1.00	1.00
16	1.00	4.08	1.01	1.00

TABLE IV: Accuracy of Visibility Estimator

Accuracy of the visibility estimator, built using MAP decoding of the belief vector at the termination of the algorithm. This table shows that the proposed algorithm is indeed learning the true visibility level during the search.

$p_V(12) = 0.2$	$p_V(12) = 0.4$	$p_V(12)0.6$	$p_V(12) = 0.8$
89.92%	75.92%	79.01%	89.35%

C. Risk of Over-Modeling

Another important issue is the potential cost of over-modeling. This is the opposite situation of Section III-A. In this case, the designer assumes the perception module is subject to the persistent factors in the environment, while in reality such factors are not significant. To investigate this issue, we use the same policies but test them in a simulated environment with independent, non-structured noise. Table V summarizes the result which suggests that it is important not to over-model.

TABLE V: Risk of Over-Modeling

The baseline model has access to the empirical error rates averaged over the prior on V. Simulations are performed for independent noise.

Single	**Time (seconds)**		**Success Rate**	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.2$	42.99	40.73	99.80%	99.85%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.4$	44.89	42.12	99.81%	99.82%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.6$	45.23	44.76	99.89%	99.77%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.8$	46.83	46.09	99.84%	99.90%
Dual	**Time (seconds)**		**Success Rate**	
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.2$	41.20	42.56	99.89%	99.71%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.4$	43.62	43.46	99.71%	99.75%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.6$	48.85	44.44	99.26%	99.78%
Method	*Visibility*	*Baseline*	*Visibility*	*Baseline*
$p_V(12) = 0.8$	58.18	45.01	99.11%	99.97%

IV. DATA COLLECTION AND PERCEPTION MODULE DESIGN

A major goal of this work is to investigate target search with drones using realistic perception modules. There is a rich and growing literature in machine learning on efficient neural networks for mobile applications [18], [23], [24]. We choose to implement a perception module based on MobileNet, in particular the variant with width multiplier of 1.0 [18]. We fine-tune from a model pre-trained on ImageNet [2], on the dataset specifically collected for this project. The particular architecture variant is selected to balance the accuracy/complexity tradeoff for our application. From benchmarking on our Samsung S8 mobile phone, the inference speed is approximately 100 ms per frame with 224×224 input images [3]. The limited size of our dataset makes it necessary for us to fine-tune on an ImageNet pretrained model. As the network are trained for 224×224 resolutions, to alleviate overfitting we design our perception module in such a way that it takes multiple cropped and/or resized images to this resolution at inference.

A. Data Collection and Model Training

As an exemplary application, we consider searching for a single basketball on the ground. To train a mobile CNN model for this application we collect image data with one or more basketballs on various locations and at different heights. In total we collect 41 clips, each around 2-3 minutes (due to the limited flying time per charge). The height ranges

[2]Available at `https://github.com/tensorflow/models/tree/master/research/slim`

[3]The phone is loaded with an Android system. Due to lack of API support, computations are performed on CPU.

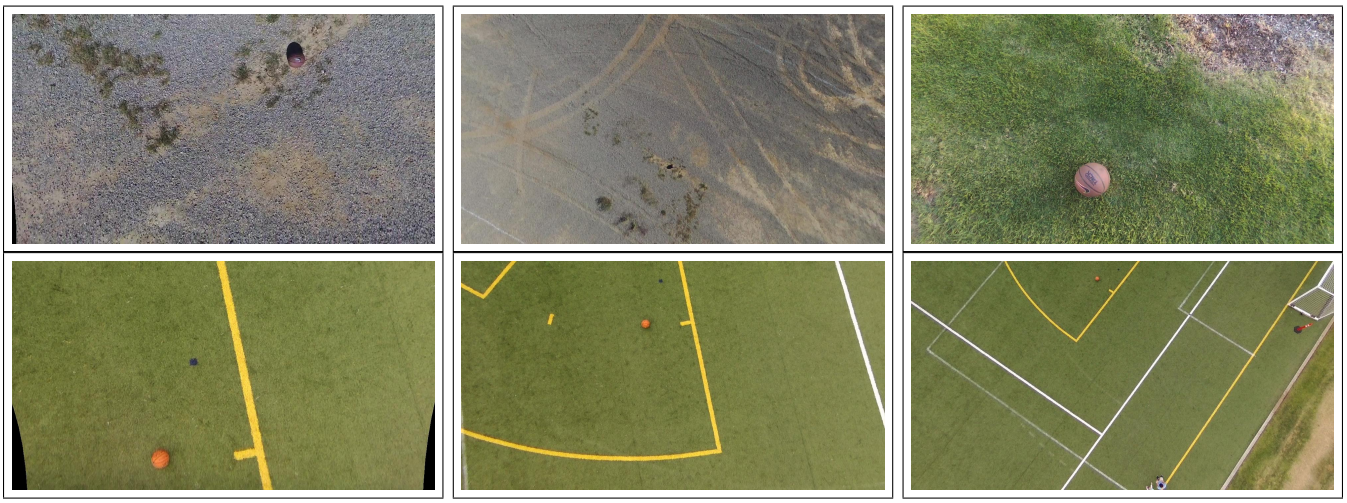

Fig. 2: Example images collected in this project. The images are collected at different heights, and with different background. The figure only shows images with basketballs. Background images without basketballs are also included in the dataset.

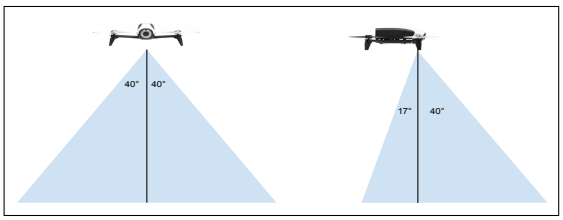

Fig. 3: The field of view of the drone.

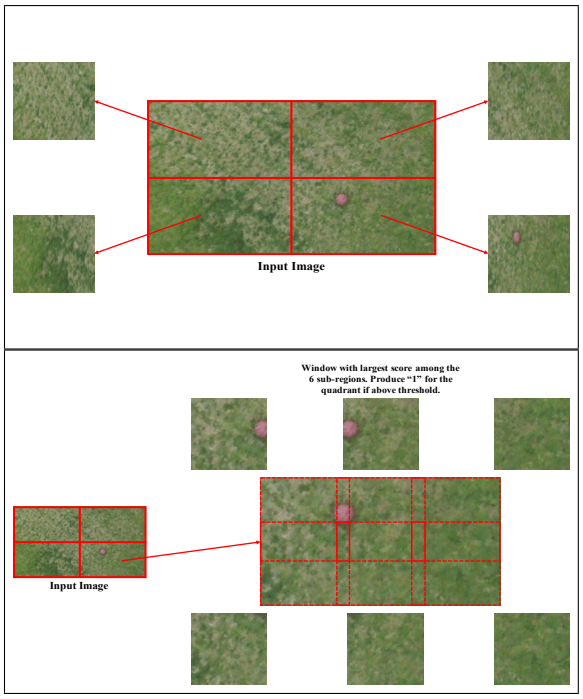

Fig. 4: Two sensing modes. **Top**: Sensing mode 1, resizing quadrants. **Bottom**: Sensing mode 2, process multiple crops at each quadrant.

from 2 meters to 64 meters. Due to safety and regulation reasons, the data collection is performed at only a few locations on and around campus with sufficient clearance

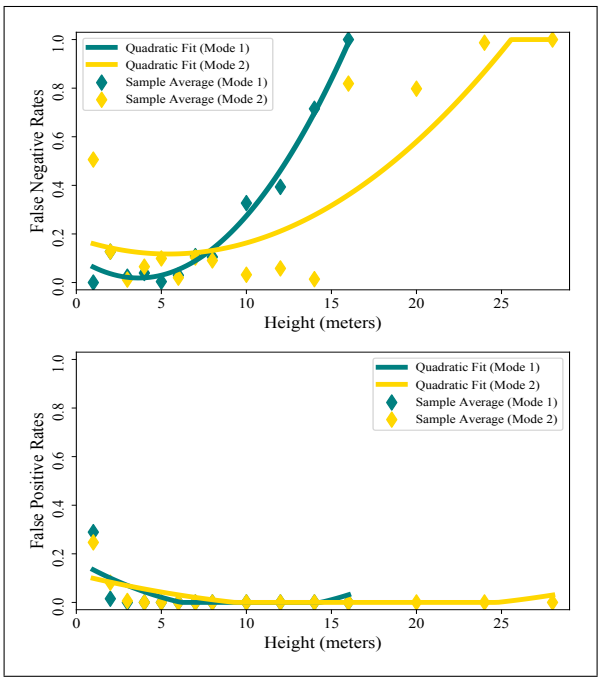

Fig. 5: False positive and false negative rates in two sensing modes. Quadratic fittings are provided to illustrate the general trends, while the original data points are summarized as scatter points.

from people, building and obstacles. There is a bias in the data collection process towards day time and good weathers. This bias is mainly due to daily routines at facilities where data collection is performed. Example images from our *basketball-search* dataset is shown in Figure 2.

The training and validation images are obtained by sampling frames extracted from the raw clips. The extracted frames are full-size 1280×720 images, the resolution of original videos saved onboard the drone. The field-of-view (FOV) of the drone is shown in Figure 3. The images are fully labeled with bounding box annotations, with 1,021

Fig. 6: Illustration of varying maximum visibility levels caused by persistent factors in the scene. All pictures are taken at 8 meters. **Left**: Image taken from the test clip at 8 meters. The image was taken at a condition in which the basketball has clear contrast with the background. Sensing mode 1 has only around 10% false negative rate on this clip. In similar conditions, sensing mode 1 will consistently produce "0" only on and above 16 meters. **Middle**: Image taken from a field test around 3pm in the afternoon. The plain background, the color of the sunlight as well as the long shadows make the basketball less visible. As a result, during repeated experiments during the field test the CNN classifier consistently reports negative ('no basketball') on and above 8 meters. **Right**: Image taken from a field test at dawn. The lighting condition is not ideal, causing large noise in the image.

positive images (w/ basketballs) and 2,304 negative images (w/o basketballs). As discussed previously during inference images are processed at 224×224 resolution, thus we use this resolution for training. The training images are random crops of 224×224 from the full-size images. Crops are assigned a binary label with the following rules: On a positive image, if a random crop overlaps with a basketball with more 80% of the area of the latter, then the crop is labeled "1"; else if its overlapping with any basketball is less than 20% of the latter its label is "0". For negative images, all crops are labeled "0". We randomly discard crops to ensure crops labeled with "1" from positive images, those labeled with "0" from positive images and crops from negative images roughly follow the a 1:1:1 proportion. The crop sampling process is repeated 4 times. This results in 4,084 crops with label "1" and 13,300 negative crops with label "0". This sampling procedure is essential in ensuring a balanced training set. A random partition separates this set into a training set and a validation set with a 3:1 ratio.

We remove the output 1000-way fully-connected layer and replace it with a 2-way output. The model is trained with softmax loss. The model is first trained for 20 epochs during which all layers except for the output layer are frozen. Then the remaining 20 epochs are trained with all layers. The fine-tuning is performed with a learning rate of 0.02. Random horizontal flipping and random saturation are added for data augmentation. The resultant model has 98.94% accuracy on training and 98.78% on validation. This CNN model is then used as the backbone for our perception module.

B. Sensing Modes and Error Characteristics

We design two sensing modes using the mobile CNN. As an image captured by the drone has a 1280×720 resolution, instead of processing it in the original resolution we partition the image into four quadrants. Each quadrant is a region with 640×360 resolutions. The outputs of the perception module are conceptually similar to the observation presented in Section II. However, instead of producing a single binary output each quadrant provides a $\{0, 1\}$ indicator of whether the particular sub-region includes the basketball. In our simulations and field tests these observations are treated as observations taken from a smaller FOV compared to the entire image. Observations from the same image are assumed independent given the target location and the maximum visibility level. We implement two sensing modes. For the less expensive sensing mode 1, we directly resize (through interpolation) to a 224×224 image. This results in less accurate observations, but the inference time is only $4 \times 100 = 400$ ms per frame. For the more expensive sensing mode 2, we perform linear scan within the 640×360 sub-regions, and take the maximum confidence score as the prediction. The test time is thus $6 \times 4 \times 100 = 2400$ ms per frame. In both test modes, the resultant confidence scores from the neural network are thresholded to obtain binary predictions. Figure 4 illustrates the two sensing modes.

We collect a separate test set from manually operating the drone at 1-meter intervals from 1 to 8 meters, 2-meter intervals from 8 to 16 meters, and 4-meter intervals from 16 to 28 meters, to empirically test the error characteristics of the proposed sensing module. Using a similar cropping strategy as in the training procedure, at each height 150 positive and negative images are collected. However, the crops in this case are 640×360, matching the size of the four quadrants of the full-size images. We then test the false positive and false negative rates at each height, of the two sensing modes. The results are summarized in Figure 5. Interestingly, different from the assumptions made at prior works [15], the errors made by the CNN-based perception module are highly imbalanced. As expected, the false negative rates (or missing rates) grow as the height increases, however false positive rates tend to decrease to near zero values at higher altitudes. Both error rates see an increase when approaching the ground. This is caused by larger targets, which results in increased chance of cases in which the basketball is partially included in the FOV.

As discussed in Section II, an important characteristic we observe from our field tests is that the predictions made by the neural network seems to have a "visibility level" structure. Figure 6 illustrates some conditions in which the

perception module fails in this way, namely consistently producing "0" even if the basketball is present during repeated tests, for all images taken above a scene dependent level. This level tends to stay constant until after changing to another field test location or perform another field test at a different time of the day. Limited by available data, it is still impossible to make statistically significant conclusions. However, this phenomenon is intuitive for the search application using drones, and has large potential impact to the search strategy. In future works we plan to investigate this further through improved data collection or through video simulation.

V. Conclusion

In this work, we investigate the problem of target search using drones. We take a real-world approach, and build a testbed that is used to investigate realistic conditions for this application. In particular, from testing perception modules using recent computer vision algorithms, we identify that persistent factors in the environment could have unexpected impact to the output of the observations. From our extensive studies through simulations, such factors have a significant influence on the design of the search algorithm. Important future directions include characterizing the impact of persistent factors more precisely, developing methods to automatically detect existence of those structures to avoid over-modeling, as well as to designing better computer vision algorithms that are more robust for target search and other related applications using drones.

References

[1] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte, "Recursive bayesian search-and-tracking using coordinated uavs for lost targets," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 2521–2526.

[2] S. Waharte and N. Trigoni, "Supporting search and rescue operations with uavs," in *2010 International Conference on Emerging Security Technologies*, Sept 2010, pp. 142–147.

[3] M. A. Goodrich, L. Lin, and B. S. Morse, "Using camera-equipped mini-uavs to support collaborative wilderness search and rescue teams," in *2012 International Conference on Collaboration Technologies and Systems (CTS)*, May 2012, pp. 638–638.

[4] D. Maturana, S. Arora, and S. Scherer, "Looking forward: A semantic mapping system for scouting with micro-aerial vehicles," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 6691–6698.

[5] K. C. T. Vivaldini, V. Guizilini, M. D. C. Oliveira, T. H. Martinelli, D. F. Wolf, and F. Ramos, "Route planning for active classification with uavs," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2563–2568.

[6] M. Popovi, G. Hitz, J. Nieto, I. Sa, R. Siegwart, and E. Galceran, "Online informative path planning for active classification using uavs," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5753–5758.

[7] M. Coombes, W. H. Chen, and C. Liu, "Boustrophedon coverage path planning for uav aerial surveys in wind," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2017, pp. 1563–1571.

[8] D. Meyer, E. Fraijo, E. Lo, D. Rissolo, and F. Kuester, "Optimizing uav systems for rapid survey and reconstruction of large scale cultural heritage sites," in *2015 Digital Heritage*, vol. 1, Sept 2015, pp. 151–154.

[9] P. Roy and V. Isler, "Active view planning for counting apples in orchards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 6027–6032.

[10] S. Waharte, A. Symington, and N. Trigoni, "Probabilistic search with agile uavs," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 2840–2845.

[11] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, "Coordinated decentralized search for a lost target in a bayesian world," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 1, Oct 2003, pp. 48–53 vol.1.

[12] T. H. Chung and J. W. Burdick, "A decision-making framework for control strategies in probabilistic search," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 4386–4393.

[13] A. Symington, S. Waharte, S. Julier, and N. Trigoni, "Probabilistic target detection by camera-equipped uavs," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 4076–4081.

[14] V. Sudevan, A. Shukla, and H. Karki, "Vision based autonomous landing of an unmanned aerial vehicle on a stationary target," in *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, Oct 2017, pp. 362–367.

[15] A. Gupta, D. Bessonov, and P. Li, "A decision-theoretic approach to detection-based target search with a uav," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 5304–5309.

[16] J. Lee, J. Wang, D. Crandall, S. abanovi, and G. Fox, "Real-time, cloud-based object detection for unmanned aerial vehicles," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, April 2017, pp. 36–43.

[17] K. R. Sapkota, S. Roelofsen, A. Rozantsev, V. Lepetit, D. Gillet, P. Fua, and A. Martinoli, "Vision-based unmanned aerial vehicle detection and tracking for sense and avoid systems," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 1556–1561.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[19] F. Vanegas, D. Campbell, M. Eich, and F. Gonzalez, "Uav based target finding and tracking in gps-denied and cluttered environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2307–2313.

[20] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee, "POMDPs for robotic tasks with mixed observability," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[21] W. S. L. Hanna Kurniawati, David Hsu, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[22] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

[23] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[24] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *CoRR*, vol. abs/1707.01083, 2017. [Online]. Available: http://arxiv.org/abs/1707.01083