Real-Time Resource Allocation for Tracking Systems

Yash Satsangi
University of Amsterdam

Shimon Whiteson
University of Oxford

Frans A. Oliehoek
University of Liverpool
University of Amsterdam

Henri Bouma
TNO, The Netherlands

Abstract

Automated tracking is key to many computer vision applications. However, many tracking systems struggle to perform in real-time due to the high computational cost of detecting people, especially in ultra high resolution images. We propose a new algorithm called *PartiMax* that greatly reduces this cost by applying the person detector only to the relevant parts of the image. PartiMax exploits information in the particle filter to select k of the n candidate *pixel boxes* in the image. We prove that PartiMax is guaranteed to make a near-optimal selection with error bounds that are independent of the problem size. Furthermore, empirical results on a real-life dataset show that our system runs in real-time by processing only 10% of the pixel boxes in the image while still retaining 80% of the original tracking performance achieved when processing all pixel boxes.

1 INTRODUCTION

Automated tracking is a key component of countless computer vision applications such as maintaining surveillance, studying traffic flows, and counting the number of people in a scene [Smeulders et al., 2014]. Consequently, in recent years many tracking systems have been proposed that make it possible to track people in a variety of challenging settings [La Cascia et al., 2000; Benfold and Reid, 2011; Smeulders et al., 2014]. However, these approaches still cannot perform real-time tracking on ultra high resolution videos (e.g., 5000×4000 pixels).

In particular, the *detection* stage, i.e., identifying an object in a scene, is the main computational bottleneck for systems that work on the *tracking-by-detection* principle [Benfold and Reid, 2011]. For example, Figure 1 shows a wide-view scene recorded by a camera mounted on top of a building [Schutte et al., 2016]. Successful tracking depends on detecting the person in the image by applying a trained detector to many *pixel boxes*. Since the scene records a wide landscape, the pixel boxes must be relatively small (e.g., 180 × 180), yielding approximately 7000 pixel boxes per image. Consequently, performing a *brute force detection* (BD) that applies the person detector to all 7000 pixel boxes is extremely computationally intensive and prohibitive to do in real time.

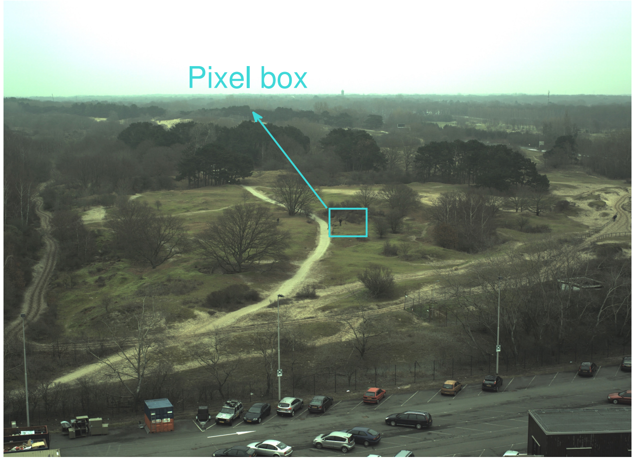

Figure 1: A wide-view scene recorded by a rooftop camera; the cyan rectangle shows an example pixel box.

In this paper, we propose a new tracking system that greatly reduces the cost of detection and thus enables real-time tracking on systems with ultra high resolution images or many cameras. The main idea is to perform *selective detection* (SD), i.e., apply the person detector not on all n pixel boxes, but only a carefully selected subset of k pixel boxes, while retaining performance guarantees, as shown in Figure 2. To do so, we build on existing techniques for *sensor selection*, which

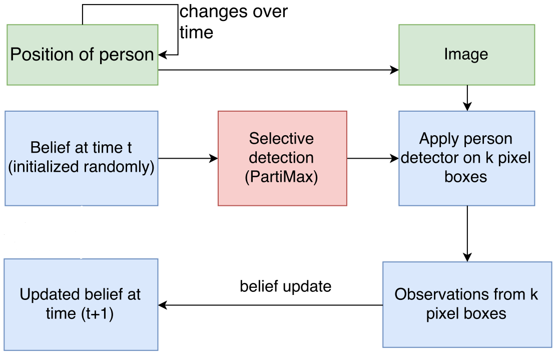
changes over time
Position of person
Image
Belief at time t (initialized randomly)
Selective detection (PartiMax)
Apply person detector on k pixel boxes
Updated belief at time (t+1)
belief update
Observations from k pixel boxes

Figure 2: Proposed tracking system with PartiMax, our proposed selective detection method, highlighted in red.

select the k out of n sensors with the highest *utility* in a multi-sensor network. Sensor selection is challenging because there are $\binom{n}{k}$ ways to perform the selection, and computing the best one would use up the same scarce computational resources we aim to intelligently allocate. Fortunately, when the utility function possesses certain characteristics, including *submodularity*, a near-optimal selection can be found using *greedy maximization*, which evaluates the utility function only $\mathcal{O}(nk)$, instead of $\binom{n}{k}$, times. In addition, *stochastic greedy maximization* [Mirzasoleiman et al., 2015] further reduces the number of evaluations of the utility function by evaluating the utility function only for a random subset of pixel boxes of size r, where $r << n$.

However, for selective detection in real-time, even stochastic greedy maximization is too expensive because computing typical utility functions such as *information gain* or *expected coverage* requires marginalizing out the observation that each candidate sensor would generate. In fact, in real-life settings with high dimensional state and/or observation spaces, evaluating information gain or expected coverage even once can be prohibitively expensive.

We start by proposing a utility function for selective detection called *particle coverage* that approximates the probability of detecting a person in a given set of pixel boxes. We show that particle coverage approximates expected coverage under certain conditions, but is much faster to compute. Then, we propose *PartiMax*. Unlike (stochastic) greedy maximization, which treats utility evaluation as a black-box, PartiMax maintains and updates the particle coverage of each pixel box in every iteration of greedy maximization, leading to large computational savings, as the particle coverage of each pixel box is not evaluated from scratch in each iteration. Furthermore, instead of selecting a subset of pixel boxes *randomly* in every iteration like stochastic greedy maximization, PartiMax samples pixel boxes with high particle coverage leading to superior tracking performance.

Since sampling pixel boxes with high particle coverage without computing the particle coverage is not trivial, we propose a sampling algorithm that is guaranteed to sample a pixel box with probability directly proportional to its particle coverage. It does so by employing *tile coding*, a popular representation in reinforcement learning that discretizes continuous spaces.

We show that, given access to a sampling algorithm like the one we propose, PartiMax is guaranteed to return a solution with tight error bounds that are independent of the problem size, i.e., independent of both n and k. Although PartiMax is designed for the particle coverage function, our bound applies generally to maximization over a set function.

Finally, we use PartiMax for selective detection to build a real-time tracking system, which we apply to a real-life dataset. Our results show that our tracking system retains 80% of its performance despite processing only 10% of each image and running in real time.

2 RELATED WORK

Most detection systems, e.g., [Felzenszwalb et al., 2010; Dollár et al., 2014], including those based on convolutional neural networks [Tian et al., 2015; Redmon et al., 2016], process the whole image and are thus not computationally efficient enough for our setting, due to the high resolution and depth of scene of the images.

Some work does identify the relevant region of interest in an image [Kim et al., 2012], e.g., by generating *proposals* (see [Hosang et al., 2015] and references therein) or saliency points [Shtrom et al., 2013]. These methods, however, are based on the properties (or low-level features) of the entire image (since they do not consider the belief state) and often generate thousands of proposals/saliency points per image. In fact, selective detection can be coupled with these methods to selectively generate saliency points.

Recently developed models of visual attention [Mnih et al., 2014; Denil et al., 2012] come closest to our work in spirit. However, they use model-free deep reinforcement learning methods to identify relevant region to apply a trained detector on [Mnih et al., 2014], while we learn the model of the world from the data and use it to plan online to find the relevant regions to which to apply a trained detector.

Our work builds off the vast existing sensor selection literature. Most work uses utility functions involving information gain [Tham and Han, 2013; Wang et al., 2005;

Satsangi et al., 2016] and expected coverage [Spaan and Lima, 2009], which are too expensive for real-time systems. Other approaches do not consider partial observability [Natarajan et al., 2012] or do not scale to large state and observation spaces [Natarajan et al., 2012; Satsangi et al., 2015]. Methods based on dynamic programming [Williams et al., 2007] or linear programming [Williams et al., 2006] or focusing on occlusions [Gupta et al., 2007] are also limited to smaller state and observation spaces.

For submodular function maximization, the most related methods are those of [Mirzasoleiman et al., 2015] and [Badanidiyuru and Vondrák, 2014]. We significantly improve upon these methods for sensor selection by introducing a novel method with lower computational cost and thereby making them applicable to real-time tracking.

3 BACKGROUND

3.1 BASIC SETUP

Let $\mathcal{X} = \{1, 2 \ldots n\}$ denote the set of all pixel boxes and i denote a single pixel box in $\mathcal{X}$. $\mathcal{A}^+$ denotes the set of all possible subsets of $\mathcal{X}$ of size less than or equal to k, $\mathcal{A}^+ = \{\mathcal{A} \subseteq \mathcal{X} : |\mathcal{A}| \leq k\}$. For the image shown in Figure 1 the size of one pixel box was chosen to be 180 x 180 pixels. The true location of the person is a hidden variable denoted by s and S is the set of all possible values s can take.[1] The observation vector $\mathbf{z} = \langle z_1, z_2 \ldots z_n \rangle$ denotes the result of applying the detector to each pixel box, i.e., each z_i denotes an estimate of whether a person appears in the pixel box i. If a pixel box i is not selected for detection, then $z_i = \emptyset$. Ω is the set of all possible values $\mathbf{z}$ can take. The *belief* $b(s)$ is a probability distribution over s. Given $\mathcal{A}$ and $\mathbf{z}$, $b(s)$ can be updated using Bayes rule.

When there are many possible states, it is not possible to maintain $b(s)$ exactly. Thus, we use particle filters, described below, to maintain and update belief $b(s)$. Below we also describe greedy maximization, which is essential to our setup, as it selects k pixel boxes out of n in $\mathcal{O}(n \times k)$ time instead of $\mathcal{O}\binom{n}{k}$.

3.2 PARTICLE FILTERS

When there are many possible states, it is infeasible to update $b(s)$ exactly. Instead, we can use *particle filters* [Doucet et al., 2001], sequential Monte Carlo algorithms for approximate inference in partially observable scenarios that are commonly used to track people in complex situations. The true belief $b(s)$ is approximated with a *particle belief* $\mathcal{B}$, a collection of m samples from $b(s)$, called particles: $\mathcal{B} = \{s_1, s_2 \ldots s_m\}$. Although weighted particle filters are often used for tracking, we use an unweighted particle filter since it can be efficiently implemented with a black-box simulator without the need to explicitly model the accuracy of the person detector or the motion dynamics.

Given the particle belief $\mathcal{B}$, a subset of sensors $\mathcal{A}$ and observation $\mathbf{z}$, particle beliefs can be updated using a *Monte Carlo belief update* [Silver and Veness, 2010]. For each particle $s_l \in \mathcal{B}$, the next state s'_l is sampled from $\Pr(s'|s)$ (under the Markov assumption) to form $\mathcal{B}' : \{s'_l : s'_l \sim \Pr(s'_l|s_l) \wedge s_l \in \mathcal{B}\}$. With enough samples, $\mathcal{B}'$ approximates the probability distribution: $b'(s') = \sum_{s \in S} \Pr(s'|s) b(s)$.

For each $s'_l \in \mathcal{B}'$, the corresponding $\mathbf{z}_l$ is drawn from $\Pr(\mathbf{z}|s'_l, \mathcal{A})$. If $\mathbf{z}_l = \mathbf{z}$, then s'_l is added to the updated belief $\mathcal{B}^{\mathcal{A}}_{\mathbf{z}}$. Otherwise, the particle is discarded. To avoid particle degeneracy, a common problem with particle filters, we combine the belief update with new particles introduced by adding random particles sampled from S to the existing particle set. $\mathcal{B}^{\mathcal{A}}_{\mathbf{z}}$ approximates the probability distribution $b^{\mathcal{A}}_{\mathbf{z}}(s') = \frac{\Pr(\mathbf{z}|\mathcal{A}, s') b'(s')}{\Pr(\mathbf{z}|b, \mathcal{A})}$.

3.3 GREEDY MAXIMIZATION

Given a set function $F(\mathcal{A})$, where $\mathcal{A} \in \mathcal{A}^+$, *greedy maximization* computes $\mathcal{A}^G$, which approximately maximizes F by building a subset of k pixel boxes iteratively. In particular, in each of its k iterations, greedy maximization adds to a partial solution the pixel box that maximizes the *marginal gain*:

$$\Delta_F(i|\mathcal{A}) = F(\mathcal{A} \cup i) - F(\mathcal{A}), \tag{1}$$

of adding i to $\mathcal{A}$, i.e., it adds $\arg\max_{i \in \mathcal{X} \setminus \mathcal{A}^G} \Delta_F(i|\mathcal{A}^G)$ to $\mathcal{A}^G$ as shown in Algorithm 1.

Algorithm 1 `greedyMax`$(F, \mathcal{X}, k)$

```
$\mathcal{A}^G \leftarrow \emptyset$
for $l = 1$ to $k$ do
    $\mathcal{A}^G \leftarrow \mathcal{A}^G \cup \arg\max_{i \in \mathcal{X} \setminus \mathcal{A}^G} \Delta_F(i|\mathcal{A}^G)$
end for
return $\mathcal{A}^G$
```

Nemhauser et al. [1978] showed that greedy maximization is guaranteed to have bounded error under certain conditions:

Theorem 1. *[Nemhauser et al., 1978] If F is non-negative, monotone and submodular, then* $F(\mathcal{A}^G) \geq$

[1] For simplicity, we sometimes assume there is only one person in the scene and the hidden variable is a vector in the Euclidean space. However, our methods and theoretical results extend easily to multiple people, as shown in Section 7.

$(1 - e^{-1})F(\mathcal{A}^*)$, *where* $\mathcal{A}^* = \arg\max_{\mathcal{A} \in \mathcal{A}^+} F(A)$.

Submodularity is a property of set functions that formalizes the notion of diminishing returns: $F : 2^{\mathcal{X}} \to \mathbb{R}$ is submodular if for every $\mathcal{A}_M \subseteq \mathcal{A}_N \subseteq \mathcal{X}$ and $i \in \mathcal{X} \setminus \mathcal{A}_N$,

$$\Delta_F(i|\mathcal{A}_M) \geq \Delta_F(i|\mathcal{A}_N). \tag{2}$$

Thus, the marginal gain of adding an element to a smaller set $\mathcal{A}_M$ is always greater than or equal to the marginal gain of adding the same element to a bigger subset $\mathcal{A}_N$ such that $\mathcal{A}_M \subseteq \mathcal{A}_N \subseteq \mathcal{X}$. If this is true for all possible values of $\mathcal{A}_N, \mathcal{A}_M$, and i, then F is submodular.

3.4 STOCHASTIC GREEDY MAXIMIZATION

Stochastic greedy maximization, shown in Algorithm 2, further reduces costs by randomly sampling a subset $\mathcal{R}$ of size r from $\mathcal{X}$ in each iteration of greedy maximization and then selecting the element from $\mathcal{R}$ that maximizes the marginal gain. It computes a subset $\mathcal{A}^S$ by adding in each iteration $\arg\max_{i \in \mathcal{R}} \Delta_F(i|\mathcal{A}^S)$, where $\mathcal{R}$ is a subset of $\mathcal{X} \setminus \mathcal{A}^S$ of size r. Mirzasoleiman et al. [2015] showed that stochastic greedy maximization is also guaranteed to have bounded error.

Algorithm 2 `stochastic-greedy-max`$(F, \mathcal{X}, k, r)$

```
$\mathcal{A}^S \leftarrow \emptyset$
for $m = 1$ to $k$ do
    $\mathcal{R} \leftarrow$ random sample of size $r$ from $\mathcal{X} \setminus \mathcal{A}^S$.
    $\mathcal{A}^S \leftarrow \mathcal{A}^S \cup \arg\max_{i \in \mathcal{R}} \Delta_F(i|\mathcal{A}^S)$
end for
return $\mathcal{A}^S$
```

Theorem 2. *[Mirzasoleiman et al., 2015] If F is non-negative, monotone and submodular, then* $\mathbb{E}[F(\mathcal{A}^S)] \geq (1 - e^{-1} - \epsilon)F(\mathcal{A}^*)$, *where* $r = \frac{n}{k}\log(\frac{1}{\epsilon})$.

3.5 UTILITY FUNCTIONS

For tracking tasks, F is often defined as *information gain* [Cover and Thomas, 1991; Krause and Guestrin, 2005; Tham and Han, 2013]:

$$IG_b(\mathcal{A}) = H_b(s) - H_b^{\mathcal{A}}(s|\mathbf{z}), \tag{3}$$

where $H_b(s)$ is the *entropy* of s and $H_b^{\mathcal{A}}(s|\mathbf{z})$ is the *conditional entropy* of s given $\mathbf{z}$ [Cover and Thomas, 1991].

It can also be defined as *expected coverage* [Spaan and Lima, 2009]. Let $\mathcal{I}_{\mathcal{B}'}^j$ be the set of particles in $\mathcal{B}'$ that are *covered* by pixel box j, $\mathcal{I}_{\mathcal{B}'}^j = \{s' \in \mathcal{B}' : j \text{ covers } s'\}$. A pixel box j covers s' if a person in state s' is visible in pixel box j. The expected coverage is defined as:

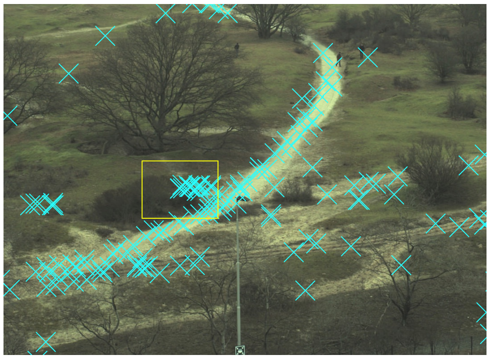

Figure 3: Particle belief: the yellow rectangle shows a pixel box and the particles it covers.

$$F_{\mathcal{B}'}(\mathcal{A}) = \sum_{\mathbf{z}} \Pr(\mathbf{z}|\mathcal{B}', \mathcal{A}) f_{\mathcal{B}_{\mathbf{z}}^{\mathcal{A}}}(\mathcal{A}), \tag{4}$$

where $f_{\mathcal{B}}(\mathcal{A}) = |\cup_{j \in \mathcal{A}} \mathcal{I}_{\mathcal{B}}^j|$. Expected coverage belongs to a general class of coverage functions that have been widely considered [Spaan and Lima, 2009]. In tracking, expected coverage is suitable because of the presence of partial observability, necessitating the expectation across $\mathbf{z}$. Expected coverage is appropriate for sensor selection or selective detection because it rewards selecting pixel boxes that have the highest probability of detecting a target. The underlying assumption is that the observations generated by the person detector are informative enough to detect a person correctly when present inside the pixel box, and are not informative enough if a person is absent from the pixel box. This is barely a restrictive assumption, as most useful person detectors satisfy it.

4 PARTICLE COVERAGE UTILITY FUNCTION

The utility functions described above are too expensive to compute in many practical settings, as they require marginalizing out observations, which is infeasible for real-time systems. In this section, we propose the *particle coverage function* (PCF) for selective detection, which does not require computing $\mathcal{B}_{\mathbf{z}}^{\mathcal{A}}$ and approximates expected coverage. PCF is defined as follows:

$$PCF_{\mathcal{B}'}(\mathcal{A}) = f_{\mathcal{B}'}(\mathcal{A}) = |\cup_{j \in \mathcal{A}} \mathcal{I}_{\mathcal{B}'}^j|. \tag{5}$$

$PCF_{\mathcal{B}'}(\mathcal{A})$ is simply the number of particles in $\mathcal{B}'$ that are covered by $\mathcal{A}$. In Figure 3, the particle coverage is the number of cyan particles that fall in the yellow pixel box. As opposed to expected coverage $F_{\mathcal{B}'}$, PCF does not involve an expectation over $\mathbf{z}$ nor does it require computing the resulting beliefs $\mathcal{B}_{\mathbf{z}}^{\mathcal{A}}$. PCF equals expected coverage under certain conditions, including the following.

Assumption 1. *For every $s' \in S$, $\mathcal{A} \subseteq \mathcal{X}$, there exist $\mathbf{z}_{s',\mathcal{A}}$ and $\bar{\mathbf{z}}_{s',\mathcal{A}}$ in Ω such that if s' is covered by $\mathcal{A}$, $\Pr(\mathbf{z}_{s',\mathcal{A}}|s', \mathcal{A}) = 1$ and if s' is not covered by $\mathcal{A}$, then $\Pr(\bar{\mathbf{z}}_{s',\mathcal{A}}|s', \mathcal{A}) = 1$.*

This assumption implies that any partial observability is due to perceptual aliasing, not noise in the sensors. Given Assumption 1, it is straightforward to show that expected coverage is equal to the particle coverage.

Theorem 3. *If Assumption 1 holds for a given $\mathcal{A}$, then $F_{\mathcal{B}'}(\mathcal{A}) = PCF_{\mathcal{B}'}(\mathcal{A})$.*

Proof. Expected coverage can be expressed as $F_{B'}(\mathcal{A}) = \sum_{\mathbf{z}\in\Omega} \Pr(\mathbf{z}|\mathcal{B}', \mathcal{A}) f_{\mathcal{B}_{\mathbf{z}}^{\mathcal{A}}}(\mathcal{A})$. In case a negative detection is observed, that is the person is not in the space covered by $\mathcal{A}$ the resulting belief will not have any particle within the space covered by $\mathcal{A}$ due to Assumption 1 and thus resulting coverage is zero. If a positive detection is observed, that is the person is inside the space covered by $\mathcal{A}$ then all the particles in resulting belief will fall within the space that is covered by $\mathcal{A}$ resulting: $f_{\mathcal{B}_{\mathbf{z}}^{\mathcal{A}}}(\mathcal{A}) = m$. This implies, $F_{B'}(\mathcal{A}) = \sum_{\mathbf{z}\in\Omega} \Pr(\mathbf{z}|\mathcal{B}', \mathcal{A})m$, where $\mathbf{z}$ is a positive observation that can be obtained only if a state is covered by $\mathcal{A}$. The probability of getting a positive detection according to $\mathcal{B}'$ is the sum of particles covered by $\mathcal{A}$ in $\mathcal{B}'$ divided by m. Thus, $F_{\mathcal{B}'}(\mathcal{A}) = \frac{PCF_{\mathcal{B}'}(\mathcal{A})}{m} \times m = PCF_{\mathcal{B}'}(\mathcal{A})$. □

In cases where Assumption 1 does not hold, particle coverage can be considered an approximation to expected coverage. Its key advantage is that computing $f_{\mathcal{B}'}$ does not require hypothetical belief updates, as one can iterate over the particle belief and simply count the number of particles that are covered by $\mathcal{A}$, making it practical for real-time applications. Moreover, it is a member of a class of coverage functions that are known to be submodular [Krause and Golovin, 2014; Takamura and Okumura, 2009] so we can employ greedy maximization to approximately maximize $f_{\mathcal{B}'}$. Our experiments show that $f_{\mathcal{B}'}$ is a good choice of utility function for selective detection in real time, leading to excellent tracking performance at a fraction of the computational cost.

Note that we formulate Assumption 1 merely for analysis purposes: to describe a set of cases in which particle coverage and expected coverage are identical. Assumption 1 is not a restrictive condition for applying PartiMax, described below. On the contrary, in the Experiments section we present excellent results for PartiMax on a real-life dataset for which Assumption 1 does not hold.

Furthermore, while we define particle coverage for the case of an unweighted particle filter, the concept is more general. In essence, the particle coverage of a pixel box is the cumulative probability mass concentrated on the states that are covered by the pixel box. Thus, any method that approximates a belief can be used to compute particle coverage by simply computing the probability mass concentrated on a set of states. For example, for a weighted particle filter, the particle coverage of a pixel box is just the sum of the weights of the particles covered by the pixel box.

5 PARTIMAX

In this section, we propose *PartiMax*, which combines the complementary benefits of PCF and stochastic greedy maximization for selective detection. Moreover, rather than merely naively applying them together, we exploit the unique structure of PCF to develop a better approach for sampling pixel boxes that is guaranteed to sample pixel boxes with high coverage, thus offering a further increase in performance. PartiMax is based on the key insight that sampling pixel boxes with a probability that is directly proportional to their particle coverage leads to strong theoretical guarantees on the expected utility. Thus, we prove error bounds for PartiMax that are independent of the number of available pixel boxes n, the number of particles in the particle filter m, or the number of pixel boxes to be selected k.

Greedy maximization and stochastic greedy maximization assume oracle access to the utility function and thus compute the marginal gain for every pixel box in every iteration. Generally, computing particle coverage function given a pixel box requires iterating over the particles to count how many fall in the space covered by the pixel box. Unlike greedy maximization, PartiMax does not explicitly compute particle coverage for each pixel box on the fly but instead maintains the particle coverage of each pixel box by updating it in every iteration. Using an approach inspired by *tile coding* [Sutton and Barto, 1998], a popular reinforcement learning technique for coding continuous state spaces, PartiMax is able to compute and maintain the particle coverage of every pixel box without having to visit n pixel boxes or m particles in every iteration.

A tile coding consists of many *tilings*. Each tiling is a set of *tiles*, which in our setting are pixel boxes. The pixel boxes in a tiling partition the state space S, i.e., they are disjoint and completely cover S. For example, Figure 4 shows two tilings in blue and yellow. Typically, different tilings have the same size pixel boxes but start at a fixed offset from each other, as in the figure. Since the pixel boxes in a given tiling form a partition, there is exactly

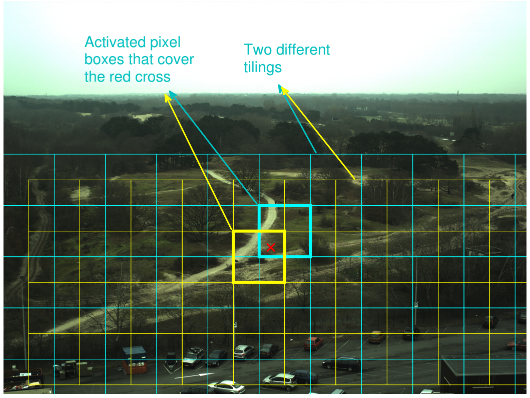

Figure 4: An example tile coding with two tilings. The highlighted tiles show the two pixel boxes that cover the red cross.

one pixel box in each tiling that covers a given state s'. If we represent each tiling as an array, locating the pixel box that covers a given state s' requires only simple arithmetic involving the size of the pixel boxes and the offset between the tilings. Figure 4 highlights the two pixel boxes, one in each tiling, that cover a given state (red cross). Thus, by representing the entire space of pixel boxes as multiple tilings, the set of pixel boxes $\mathcal{T}_{s'}$ that covers a given state s' can be identified in constant time.

In reinforcement learning, tile codings are used to discretize continuous state spaces in order to approximate a value function. Here, we use it differently, just as a scheme for dividing an image into overlapping pixel boxes. The benefit of this approach is that it enables PartiMax to maintain Δ_f efficiently, by providing constant-time access to the set $\mathcal{T}_{s'}$ of all pixel boxes that cover a given state s', i.e., $\mathcal{T}_{s'} = \{i \in \mathcal{X} : i \text{ covers } s'\}$.

Algorithm 3 `PartiMax`$(\mathcal{B}', \mathcal{X}, k)$

```
$\langle \Phi, \Delta_f \rangle \leftarrow$ initialize$(\mathcal{B}', \mathcal{X})$
$\mathcal{A}^S \leftarrow \emptyset$.
for $l = 1$ to $k$ do
    $\mathcal{R} \leftarrow$ sampleP$(r, \mathcal{B}', \mathcal{X}, \mathcal{A}^S)$
    $i' \leftarrow \arg\max_{i \in \mathcal{R}} \Delta_f(i|\mathcal{A}^S)$.
    $\mathcal{A}^S \leftarrow \mathcal{A}^S \cup i'$
    $\langle \Delta_f, \Phi \rangle \leftarrow$ update$(\Delta_f, \Phi, i', \mathcal{A}^S, \mathcal{X})$
end for
return $\mathcal{A}^S$
```

Algorithm 3 shows pseudocode for PartiMax. It starts by calling `initialize` (Algorithm 4), which returns two data structures, Φ and Δ_f. $\Phi(i|\emptyset)$ stores for each i the set of particles in $\mathcal{B}'$ that i covers; and $\Delta_f(i|\emptyset)$ is the number of particles that are covered by i. For each particle $s' \in \mathcal{B}'$, `initialize` calls `covers`, which uses the tile coding to find the set of pixel boxes $\mathcal{T}_{s'}$ that cover that particle. For every activated pixel box, $i \in \mathcal{T}_{s'}$, $\Delta_f(i|\emptyset)$ is incremented and s' is added to the set of particles $\Phi(i|\emptyset)$.

Algorithm 4 `initialize`$(\mathcal{B}', \mathcal{X})$

```
$\Phi(i|\emptyset) \leftarrow \emptyset \forall\, i \in \mathcal{X}$
$\Delta_f(i|\emptyset) \leftarrow 0 \;\forall\, i \in \mathcal{X}$
for $s' \in \mathcal{B}'$ do
    $\mathcal{T}_{s'} \leftarrow$ covers$(s')$
    $\Phi(i|\emptyset) \leftarrow \Phi(i|\emptyset) \cup \{s'\} \;\forall i \in \mathcal{T}_{s'}$
    $\Delta_f(i|\emptyset) = \Delta_f(i|\emptyset) + 1 \;\forall i \in \mathcal{T}_{s'}$
end for
return $\langle \Phi, \Delta_f \rangle$
```

Once Φ and Δ_f are returned by `initialize`, PartiMax proceeds like stochastic greedy maximization, adding in each iteration the pixel box i' that maximizes the marginal gain from $\mathcal{R}$. Since going over all pixel boxes is too expensive, PartiMax calls Algorithm 5 to obtain $\mathcal{R}$, a subset of $\mathcal{X}$ of size r ($r << n$). However, unlike stochastic greedy maximization, $\mathcal{R}$ is not sampled uniformly randomly but instead Algorithm 5 samples from a distribution such that the probability that i is included in $\mathcal{R}$ is directly proportional to the particle coverage of i.

Algorithm 5 `sampleP`$(r, \mathcal{B}', \mathcal{X}, \mathcal{A}^S)$

```
$\mathcal{R} \leftarrow \emptyset$
while $|\mathcal{R}| < r$ do
    $s' \sim \text{Unif}(\mathcal{B}')$
    if $s'$ is not covered by $\mathcal{A}^S$ then
        $\mathcal{T}_{s'} \leftarrow$ covers$(s')$
        $i \sim \text{Unif}(\mathcal{T}_{s'})$ // uniformly random sample from $\mathcal{T}_{s'}$
        $\mathcal{R} \leftarrow \mathcal{R} \cup i$
    end if
end while
return $\mathcal{R}$
```

Algorithm 6 `update`$(\Delta_f, \Phi, i', \mathcal{A}^S, \mathcal{X})$

```
for $s' \in \Phi(i')$ do
    $\mathcal{T}_{s'} \leftarrow$ covers$(s')$
    $\Delta_f(i|\mathcal{A}^S) = \Delta_f(i|\mathcal{A}^S) - 1 \;\forall i \in \mathcal{T}_{s'}$
    $\Phi(i|\mathcal{A}^S) \leftarrow \Phi(i|\mathcal{A}^S) \setminus s' \;\forall i \in \mathcal{T}_{s'}$
end for
return $\langle \Delta_f, \Phi \rangle$
```

In general, sampling from such a distribution would be difficult, but with PCF we can do this efficiently. Algorithm 5 first uniformly randomly samples a particle from the belief. If the particle is not covered by $\mathcal{A}^S$, then it uses tile coding to find the set of pixel boxes $\mathcal{T}_{s'}$ that cover s' and adds a pixel box uniformly randomly from $\mathcal{T}_{s'}$. This is repeated until r pixel boxes are added to $\mathcal{R}$.

At the end of each iteration, PartiMax calls update (Algorithm 6), which updates $\Delta_f(i|\mathcal{A}^S)$ and $\Phi(i|\mathcal{A}^S)$ for every $i \in \cup_{s' \in \Phi(i'|\mathcal{A}^S)} \mathcal{T}_{s'}$. It starts by iterating over the particles s' in $\Phi(i'|\mathcal{A}^S)$ and for each particle uses the tile coding to find $\mathcal{T}_{s'}$. For every pixel box $i \in \mathcal{T}_{s'}$, $\Delta_f(i|\mathcal{A}^S)$ is decremented and s' is removed from $\Phi(i|\mathcal{A}^S)$, to account for the fact that i' now covers s' and thus the marginal gain of i is reduced. The marginal gain of every other i remains unchanged. Similarly, $\Phi(i|\mathcal{A}^S)$ is updated by subtracting s' from $\Phi(i|\mathcal{A}^S)$ for every i in $\mathcal{T}_{s'}$.

6 ANALYSIS

We now establish bounds on the cumulative error of PartiMax that are independent of the problem size. We start with a lemma that shows that the probability of adding i to $\mathcal{R}$ via Algorithm 5 is directly proportional to the marginal gain of i.

Lemma 1. *Let* $i = \texttt{sampleP}(1, \mathcal{B}', \mathcal{X}, \mathcal{A}^S)$ *then* $\Pr_{\mathcal{A}^S}(i = i_j) = c\Delta(i_j|\mathcal{A}^S)$, *where* $c = \frac{1}{tm'}$ *is a constant where* t *is the number of tilings and* m *is the number of particles in* $\mathcal{B}'$ *that are not covered by* $\mathcal{A}^S$.

Proof. The probability that a given pixel box i_j is sampled in $\mathcal{T}_{s'}$ (in line 5) is the number of particles covered by i_j in $\mathcal{B}'$ that are not covered by $\mathcal{A}^S$, which is $\Delta_f(i_j|\mathcal{A}^S)$:

$$\Pr(i_j \in \mathcal{T}_{s'}) = \frac{1}{m}\Delta_f(i_j|\mathcal{A}^S). \tag{6}$$

Since there is exactly one pixel box that covers a given state in each of the t tilings, the total number of pixel boxes, that is the size of $\mathcal{T}_{s'}$ is t. Since Algorithm 5 samples uniformly randomly from $\mathcal{T}_{s'}$, then the probability of selecting i_j from $\mathcal{T}_{s'}$ is $\frac{1}{|\mathcal{T}_{s'}|} = \frac{1}{t}$. Thus,

$$\Pr(i = i_j) = \frac{1}{t}\frac{1}{m}\Delta_f(i_j|\mathcal{A}^S). \tag{7}$$

□

Next, we show that PartiMax is guaranteed to be near-optimal.

Theorem 4. *Let* F *be a set function over a collection of sets* $\mathcal{A}^+ = \{\mathcal{A}_1, \mathcal{A}_2 \dots \mathcal{A}_v\}$ *and let* $\mathcal{A}^* = \arg\max_{\mathcal{A} \in \mathcal{A}^+} F(\mathcal{A})$, *let* $\mathcal{A}' = \arg\max_{\mathcal{A} \in \mathcal{R}} F(\mathcal{A})$, *such that* $\mathcal{R}$ *is formed by sampling* r *sets from a probability distribution such that probability of sampling* $\mathcal{A}$ *is* $\Pr(\mathcal{A}) = \frac{1}{c}F(\mathcal{A})$, *where* c *is a scalar constant, such that,* $\frac{c}{F(\mathcal{A}^*)} - 1 \leq r$. *Then,*

$$F(\mathcal{A}^*) - \mathbb{E}F(\mathcal{A}') \leq (\frac{r}{1+r})^r F(\mathcal{A}^*). \tag{8}$$

Proof. Let $p_1, p_2 \dots p_v$ denote $P(\mathcal{A}_1), P(\mathcal{A}_2) \dots P(\mathcal{A}_v)$ respectively. Also without loss of generality, we assume $p_1 \geq p_2 \geq \dots p_v$. Consequently, it follows, $F(\mathcal{A}_1) \geq F(\mathcal{A}_2) \geq \dots \geq F(\mathcal{A}_v)$. Note $\mathcal{A}^* = \mathcal{A}_1$. The expected value of $F(\mathcal{A}')$ is at least as much as:

$$\mathbb{E}[F(\mathcal{A}')] \geq (1 - (1 - p_1)^r)F(\mathcal{A}_1). \tag{9}$$

The term on the right corresponds to the case, when $\mathcal{A}_1$ is sampled at least once in $\mathcal{R}$, then we are guaranteed to get $\mathcal{A}' = \mathcal{A}_1$. The rest of the cases when $\mathcal{A}_1$ is not sampled in $\mathcal{R}$, we ignore, thus giving us the above bound on $\mathbb{E}[F(\mathcal{A}')]$. Thus,

$$F(\mathcal{A}_1) - \mathbb{E}[F(\mathcal{A}')] \leq F(\mathcal{A}_1) - (1 - (1 - p_1)^r)F(\mathcal{A}_1).$$

Since $cp_1 = F(\mathcal{A}_1)$, the above equation can be written as:

$$F(\mathcal{A}_1) - \mathbb{E}[F(\mathcal{A}')] \leq cp_1 - (1 - (1 - p_1)^r)cp_1. \tag{10}$$

On differentiating the right hand side with respect to p_1 and equating it to zero, we find that the maxima of right hand size occurs at $p_1 = (1/(r+1))$. Also, since $r \geq \frac{c}{F(\mathcal{A}^*)} - 1$, substituting this in the above equation we get,

$$F(\mathcal{A}_1) - \mathbb{E}[F(\mathcal{A}')] \leq (r/(r+1))^r F(\mathcal{A}_1).$$

□

The above theorem guarantees that, granted access to a probability distribution such that $\Pr(\mathcal{A}) = cF(\mathcal{A})$, there exists a tight theoretical guarantee for selecting $\mathcal{A}' = \arg\max_{\mathcal{A} \in \mathcal{R}} F(\mathcal{A})$, independent of the problem size.

Directly applying Theorem 4 and Lemma 1 yields the following lemma, which shows that the marginal gain of PartiMax $\Delta_f(i'|\mathcal{A}^S)$ in each iteration is at least $(1 - (\frac{r}{r+1})^r)\Delta_f(i^*|\mathcal{A}^S)$, where $i^* = \arg\max_{i \in \mathcal{X} \setminus \mathcal{A}^S} \Delta(i|\mathcal{A}^S)$ and $r \geq \frac{tm}{2} - 1$.

Lemma 2. *Let* $i^* = \arg\max_{i \in \mathcal{X} \setminus \mathcal{A}^S} \Delta_f(i|\mathcal{A}^S)$, $r \geq \frac{tm}{2} - 1$ *and let* $i' = \arg\max_{i \in \mathcal{R}} \Delta_f(i|\mathcal{A}^S)$, *where* $\mathcal{R} = \texttt{sampleP}(\texttt{r}, \mathcal{B}', \mathcal{X}, \mathcal{A}^\text{S})$, $r \geq \frac{tm}{2} - 1$, *then,*

$$\Delta_f(i^*|\mathcal{A}^S) - \mathbb{E}\Delta_f(i'|\mathcal{A}^S) \leq (\frac{r}{r+1})^r \Delta_f(i^*|\mathcal{A}^S).$$

Proof. Using $\mathcal{X} \setminus \mathcal{A}^S$ as $\mathcal{A}^+$, i^* as $\mathcal{A}^*$, i' as $\mathcal{A}'$ and applying Theorem 4 and Lemma 1 yields the desired result. □

Lemma 2 in turn yields the following theorem for $r \geq \frac{tm}{2} - 1$:

Theorem 5.

$$\mathbb{E}[f(\mathcal{A}^S)] \geq (1 - e^{-1} - (r/(r+1)^r))f(\mathcal{A}^*). \tag{11}$$

Proof. Let $\mathcal{A}^* = \{i_1^*, i_2^*, \dots, i_k^*\}$ and $\mathcal{A}_m^S = \{i_1^S, i_2^S, \dots, i_m^S\}$ be the solution returned by PartiMax after $m \leq k$ iterations. Let $\mathcal{A}^P = \mathcal{A}^* \setminus \mathcal{A}_m^S = \{i_1^P, \dots, i_j^P\}$ and let $\mathcal{A}_l^P$ be the first l elements of $\mathcal{A}^P$, with $\mathcal{A}_0^P = \emptyset$. Note that $\mathbb{E}f(\mathcal{A}^* \cup \mathcal{A}_m^S)$ can be expressed as:

$$\mathbb{E}[f(\mathcal{A}^* \cup \mathcal{A}_m^S)] = \mathbb{E}[f(\mathcal{A}_m^S)] + \sum_{l=1}^{j} \mathbb{E}[\Delta_f(i_l^P | \mathcal{A}_m^S \cup \mathcal{A}_{l-1}^P\})].$$

f is monotonic, $\mathbb{E}[f(\mathcal{A}^* \cup \mathcal{A}_m^S)] \geq \mathbb{E}[f(\mathcal{A}^*)]$, and by submodularity, $\sum_{l=1}^{j} \mathbb{E}[\Delta_f(i_l^P | \mathcal{A}_m^S)] \geq \sum_{l=1}^{j} \mathbb{E}[\Delta_f(i_l^P | \mathcal{A}_m^S \cup \mathcal{A}_{l-1}^P\})]$. Thus,

$$\mathbb{E}[f(\mathcal{A}_m^S)] + \sum_{i \in \mathcal{A}^P} \mathbb{E}[\Delta_f(i | \mathcal{A}_m^S)] \geq f(\mathcal{A}^*). \tag{12}$$

From Lemma 2, $\mathbb{E}[f(\mathcal{A}_{m+1}^S) - f(\mathcal{A}_m^S)] \geq \Delta_f(i^* | \mathcal{A}_m^S) - (\frac{r}{r+1})^r \Delta_f(i^* | \mathcal{A}_m^S)$. Also, since $|\mathcal{A}^P| \leq k$,

$$\begin{aligned} \mathbb{E}[f(\mathcal{A}_{m+1}^S) - f(\mathcal{A}_m^S)] + (\frac{r}{r+1})^r \Delta_f(i^* | \mathcal{A}_m^S)) \\ \geq \frac{1}{k}[f(\mathcal{A}^*) - \mathbb{E}[f(\mathcal{A}_m^S)]]. \end{aligned} \tag{13}$$

By induction on m the desired result can be obtained. ([Krause and Golovin, 2014; Mirzasoleiman et al., 2015; Satsangi et al., 2015]) □

The above theorem establishes a bound on the error of PartiMax that is independent of the size of the problem and thus remains tight even for large values of n. Furthermore, the above result shows that, as the size of $\mathcal{R}$ increases, PartiMax's performance is guaranteed to converge to that of greedy maximization.

While we have shown these results for PartiMax for selective detection, Theorem 4 is applicable to any problem that involves maximization over a set function where we can sample from a probability distribution such that the probability of sampling a subset $\mathcal{A}$ is directly proportional to the value of that subset specified by the set function F. Also note that Theorem 4 does not make any assumptions about F and is applicable to any set function, submodular or not.

7 EXPERIMENTS

We evaluated PartiMax on a dataset containing approximately 2100 trajectories of people recorded by a camera taking 5120×3840 resolution images running at 6 frames per second [Schutte et al., 2016]. The trajectories were generated using the *ACF* detector [Dollár et al., 2014] and in-camera tracking [Schutte et al., 2016]. These tracks were used to learn the *motion model* of the people walking in the scene, as described below.

We model the state s as the person's position and velocity, $s = \langle x, y, v_x, v_y \rangle$, where x and y describe position and v_x and v_y describe velocity. Both x and y are integers in $\{0, \dots, 5000\}$. We use a motion model that predicts the next position as:

$$x_{next} = x_{curr} + v_x^{curr} + \mathcal{N}(0, \sigma_x), \tag{14}$$

for x and analogously for y. We use a maximum likelihood estimate of σ_x learned from the data.

Each pixel box was 180×180 and each tiling had a 60×30 offset from the previous one. This offset was chosen because it is the size of the average *bounding box* required to bound a detected person in the scene. This setup yields approximately 7000 pixel boxes per image.

We assume access to a detector that determines with 90% accuracy whether a person is located within a given pixel box and gives a noisy observation about the location of the person if detected. Using the motion model and this detector, we maintain a particle belief $\mathcal{B}$ about the person's location using an unweighted particle filter with 250 particles. Multi-person tracking uses a separate particle filter for each person.

In our experiments, each algorithm selects k pixel boxes to which to apply the detector. To evaluate its performance, we sample a test trajectory from the dataset and try to track the person's movement, starting with a random belief $\mathcal{B}$ and updating it at each timestep using the observations generated from the selected pixel boxes. At each timestep, the agent is asked to predict the position of the person in the scene and gets a reward of +1 for correct predictions and 0 otherwise. Performance is quantified as the total cumulative reward aggregated by the agent at the end of a trajectory over a series of 50 timesteps.

The experiments were run for over 140 trajectories for 8 independent runs for 1 person tracking and 6 independent runs for 3 and 5 person tracking.

As a baseline, we compare against an efficient version of greedy maximization (GM+PCF) (in red in plots) that employs tile coding to maintain the particle coverage of each pixel box. GM+PCF is the same as PartiMax but, instead of selecting the pixel box with the highest particle coverage, in each iteration from $\mathcal{R}$, GM+PCF selects it from $\mathcal{X}$. A naive implementation of greedy maximization that computes the particle coverage of each pixel box in every iteration by going over the entire belief was too slow for a complete run and required 160 seconds to select $k = 40$ from $n = 7200$ for one-person tracking. GM+PCF returns the same solution as greedy maximization but is faster. Simple baselines like downsampling are

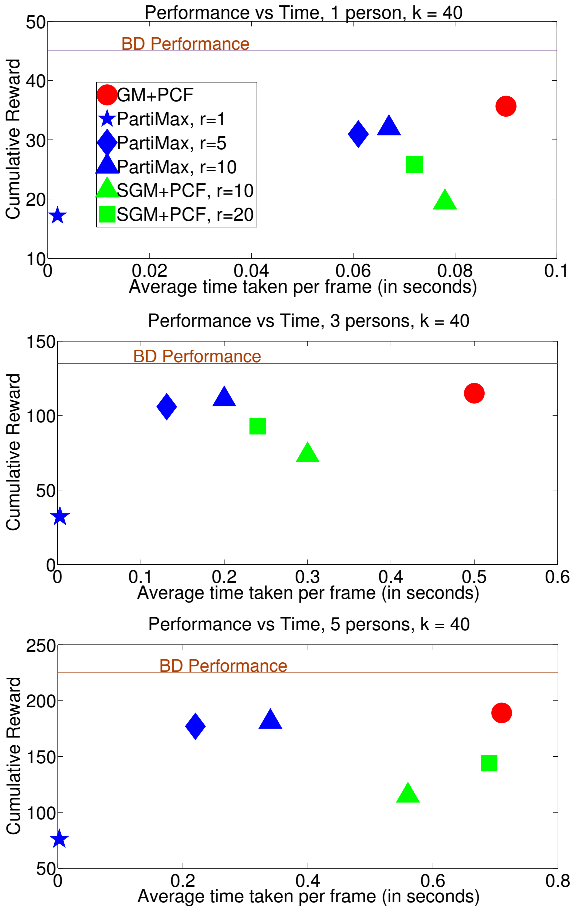

Figure 5: Total correct predictions vs. CPU time (seconds) for tracking one (top), three (middle) and five (bottom) people. The closer to the top-left corner, the better.

not useful as the tracking system must still process thousands of pixel boxes even if the image is downsampled by a factor of 4 or 8. Furthermore, downsampling precludes detection of high level features about the person like the color of his/her clothes, etc., thus defeating the purpose of deploying a high resolution camera.

We also compare to stochastic greedy maximization (in green in the plots) that randomly samples a subset $\mathcal{R}$ from $\mathcal{X}$ but employs tile coding to maintain the particle coverage of each pixel box. A naive implementation of stochastic greedy maximization that computes particle coverage of each pixel box from scratch takes around 0.83 seconds for $k = 40$ and $r = 10$ for one person tracking. The combination of SGM + PCF returns the same solution as stochastic greedy maximization, but faster.

Figure 5 shows a detailed comparison between PartiMax, greedy maximization, and stochastic greedy maximization when tracking 1, 3, or 5 people with $k = 40$. The y-axis shows the cumulative correct predictions averaged over multiple trajectories that the agent made using observations from each algorithm and the x-axis shows the time taken by each algorithm to select 40 out of 7200 pixel boxes. Thus, the top left corner indicates good tracking performance at a low computational cost. The brown line in the figure shows the tracking performance when the brute force detection is used, that is the person detector is applied to the entire image (except the part containing sky), which takes approximately 2.5 seconds.

The blue diamond and triangle at the top left corner of each plot show the superior performance and computational efficiency of PartiMax compared to the baselines. PartiMax not only matches the performance of greedy maximization, it does so extremely efficiently with a low value of r, thanks to the sampling scheme we propose. Stochastic greedy maximization's tracking performance suffers due to its random sampling, while the computational cost of GM+PCF increases with the number of people. PartiMax combines the best of both of these baselines and performs better both in terms of tracking performance and computational cost. In fact, as the number of people in the scene increases, PartiMax scales much better than any other algorithm. Overall, PartiMax is able to retain 80% percent of BD's tracking performance but is at least 10 times faster.

8 CONCLUSIONS & FUTURE WORK

This paper proposed a new tracking system that selectively processes only a fraction of an image to track people in real time. We proposed a new algorithm PartiMax that exploits submodularity to quickly identify the most relevant regions in an image. We applied our tracking system to a real-life dataset and showed that it retains 80% of tracking performance even while processing only a fraction of each image and running in real time. In future we plan to apply PartiMax to sensor selection tasks and other applications that involves maximizing coverage functions.

Acknowledgements

We thank TNO for providing us with the dataset used in our experiments. We also thank the STW User Committee for its advice regarding active perception for tracking systems. This research is supported by the Dutch Technology Foundation STW (project #12622), which is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs. Frans Oliehoek is funded by NWO Innovational Research Incentives Scheme Veni #639.021.336.

References

A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *ICML*, 2014.

B. Benfold and I. Reid. Stable multi-target tracking in real-time surveillance video. In *CVPR*, 2011.

T.M. Cover and J.A. Thomas. *Entropy, relative entropy and mutual information*. Wiley-Interscience, 1991.

M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas. Learning where to attend with deep architectures for image tracking. *Neural computation*, 2012.

P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *TPAMI*, 36(8), 2014.

A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2001.

P. Felzenszwalb, Girshick, D McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9), 2010.

A. Gupta, A. Mittal, and L.S. Davis. Cost: An approach for camera selection and multi-object inference ordering in dynamic scenes. In *ICCV*, 2007.

J. Hosang, M. Omran, R. Beneson, and B. Schiele. Taking a deeper look at pedestrians. In *CVPR*, 2015.

K. Kim, D. Lee, and I. Essa. Detecting regions of interest in dynamic scenes with camera motions. In *CVPR*. IEEE, 2012.

A. Krause and D. Golovin. Submodular function maximization. Cambridge University Press, 2014.

A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI*, 2005.

M. La Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3D models. *TPAMI*, 2000.

B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause. Lazier than lazy greedy. In *AAAI*, 2015.

V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *NIPS*, 2014.

P. Natarajan, T.N. Hoang, K.H. Low, and M. Kankanhalli. Decision-theoretic approach to maximizing observation of multiple targets in multi-camera surveillance. In *AAMAS*, 2012.

G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14, 1978.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: unified real-time object detection. 2016.

Y. Satsangi, S. Whiteson, and F. Oliehoek. Exploiting submodular value functions for faster dynamic sensor selection. In *AAAI*, 2015.

Y. Satsangi, S. Whiteson, and F. Oliehoek. PAC greedy maximization with efficient bounds on information gain for sensor selection. In *IJCAI 2016*, July 2016.

K. Schutte, G. Burghouts, N. Van der Stap, V. Westerwoudt, et al. Long-term behavior understanding based on the expert-based combination of short-term observations in high-resolution CCTV. In *SPIE*, volume 9995, 2016.

E. Shtrom, G. Leifman, and A. Tal. Saliency detection in large point sets. In *ICCV*, 2013.

D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *NIPS*, 2010.

A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *TPAMI*, 2014.

M.T.J. Spaan and P.U. Lima. A decision-theoretic approach to dynamic sensor selection in camera networks. In *ICAPS*, 2009.

R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

H. Takamura and M. Okumura. Text summarization model based on maximum coverage problem and its variant. In *Conf. Europ. Chapter Assoc. Comp. Ling.*, 2009.

C.-K. Tham and M. Han. Information-driven sensor selection for energy-efficient human motion tracking. In *IEEE Int. Conf. Distr. Comp. in Sensor Syst.*, 2013.

Y. Tian, P. Luo, X. Wang, and X. Tang. Pedestrian detection aided by deep learning semantic tasks. In *CVPR*, 2015.

H. Wang, K. Yao, and D. Estrin. Information-theoretic approaches for sensor selection and placement in sensor networks for target localization and tracking. *IEEE J. Comm. and Networks*, 2005.

J.L. Williams, J.W. Fisher III, and A.S. Willsky. Sensor management for multiple target tracking with heterogeneous sensor models. In *SPIE*, volume 6235, 2006.

J.L. Williams, J.W. Fisher, and A.S. Willsky. Approximate dynamic programming for communication-constrained sensor network management. *IEEE Trans. Signal Proc.*, 55(8), 2007.