WiseCam: Wisely Tuning Wireless Pan-Tilt Cameras for Cost-Effective Moving Object Tracking

Jinlong E*†, Lin He†‡, Zhenhua Li†, Yunhao Liu†
* Renmin University of China † Tsinghua University ‡ Zhongguancun Laboratory, China
{*ejinlong89, helin1170, lizhenhua1983, yunhaoliu*}*@gmail.com*

***Abstract*—With desired functionality of moving object tracking, wireless pan-tilt cameras are able to play critical roles in a growing diversity of surveillance environments. However, today's pan-tilt cameras oftentimes underperform when tracking frequently moving objects like humans – they are prone to lose sight of objects and bring about excessive mechanical rotations that are especially detrimental to those energy-constrained outdoor scenarios. The ineffectiveness and high cost of state-of-the-art tracking approaches are rooted in their adherence to the industry's simplicity principle, which leads to their stateless nature, performing gimbal rotations based only on the latest object detection. To address the issues, we design and implement WiseCam that wisely tunes the pan-tilt cameras to minimize mechanical rotation costs while maintaining long-term object tracking. We examine the performance of WiseCam by experiments on two types of pan-tilt cameras with different motors. Results show that WiseCam significantly outperforms the state-of-the-art tracking approaches on both tracking duration and power consumption.**

I. Introduction

As one of the most important professional surveillance instruments, wireless pan-tilt cameras have gained tremendous popularity in recent years. Their overall global market value has exceeded $3 billion in 2020 and is expected to steadily grow at ∼2.5% in the next years [1]. At present, pan-tilt cameras manufactured by dominant companies (*e.g.,* Axis, FLIR, and Honeywell) almost target DC electricity-supplied indoor monitoring scenarios such as elderly/child caring and anti-theft alarming, where the most desired functionality is *moving object tracking*. Owing to wider-area coverage brought by the flexible rotations, pan-tilt cameras are also potentially conducive to surveillance in diverse rural off-the-grid environments (*e.g.,* farmlands, fisheries), in replace of multiple commonly-used fixed cameras.

Nevertheless, the tracking performance of today's pan-tilt cameras is far from satisfactory, especially for frequently moving objects like humans and in the energy-constrained scenarios (§II-A). Most commodity pan-tilt cameras operate with two stepper motors, each of which drives a gear shaft to rotate grid by grid [2]. Typically, they adopt a *grid-based tracking* approach that continuously seeks a moving object through *vision detection* and then searches for pan-tilt grids that makes the object closest to the camera's gaze direction. Unfortunately, the two-dimensional searching process involves time-consuming rotations and comparisons among grids, making the camera hardly keep sight of object motion. Meanwhile, frequent acceleration and deceleration among grids incur large power consumption.

There are also pan-tilt cameras assembled with a small fixed camera and an electric tripod head driven by two servo motors, which conduct rotations by any angles with high speed and precision [3]. The mechanical properties make it suitable to the *target-based tracking* approach. Whenever a detected object is not in the center of the camera's field of view[1] (termed *view center*), pan-tilt rotating angles are calculated based on the object center's two-dimensional relative coordinates, respectively [4], [5]. Such an approach is free from inefficient grid search, but it sensitizes the camera to the object's small motions and results in excessive stacked rotations. The camera consumes a large amount of energy and probably loses sight of objects when asynchronously executing them.

An intuitive approach to overcoming the above obstacle is to set a fault-tolerant boundary around the camera's view center and restrains pan-tilt rotations when the target object's center moves within it [6]. As the boundary size is subject to a number of object properties (*e.g.,* size, speed, direction), it is difficult to set a proper one in practice – either too small to reduce the frequency of rotation generation or too large to catch up with the object. An alternative approach to reducing the camera's rotation delays is to adopt a proportion-integration-differentiation (PID) controller [7] that iteratively calculates the difference between the object center and view center as an error value and obtains pan-tilt rotating angles based on correction of PID terms. However, complex tuning of PID coefficients is required to satisfy the validity criteria.

In a nutshell, all the state-of-the-art approaches continuously conduct object detection and statelessly determine pan-tilt rotations only based on an object's instant position, which leads to considerable energy costs and a high risk of tracking failure (§II-B). The stateless nature of these tracking approaches mainly results from the industry's simplicity principle [8] (*i.e.,* generally adopting those easy-to-implement and low-overhead schemes), but it is often inapplicable to complex environments and/or high demands in practice. In view of this, we design WiseCam to minimize pan-tilt cameras' rotation costs while maintaining its long-term object tracking online (§III-A). Our key idea is to follow closely state variations of the target object and accordingly refrain from dispensable gimbal rotations.

[1]The area that is visible to a camera is normally referred to as the camera's field of view.

While the idea is promising and straightforward, to make it work in high efficiency and with low overhead, we focus on addressing the following two technical challenges.

- *To keep a close watch on an object in a unified space*, we ameliorate the pan-tilt cameras' detection scheme and integrate *correlation filtering* that matches the detected object in consecutive frames by element-wise operations on multiple features. On this basis, we abstract critical points from numerous object pixels in each frame and transform their non-unified coordinates to the trajectory of geodetic coordinates in a panoramic space (§III-B).
- *To efficiently learn object motion data for online determination*, we design a *fast-convergent* reinforcement learning (RL) model to tune the amplitude of pan-tilt rotations, with multiple associated neural networks and well-customized rewards to capture spatio-temporal motion features. We further explore learning experience from different objects of the same type (*e.g.,* humans) and fuse them by model aggregation, so as to accelerate generalization for determination on new objects (§III-C).

We implement WiseCam to support moving object tracking on both stepper-driven and servo-driven pan-tilt cameras (§IV-A). We evaluate WiseCam on the two heterogeneous cameras through real-world human tracking experiments and data-driven testing of walking trajectories involving thousands of humans (§IV-B∼§IV-D). In contrast to the state-of-the-art tracking approaches, WiseCam yields 2∼8 times longer average tracking duration, and meanwhile, it can reduce the rotational power consumption of the test cameras by at least 38% and is applicable to the solar-powered pan-tilt cameras. In addition, the overhead of WiseCam keeps low and steady when confronted with intensive tracking tasks (the resource usages are steady at 5%∼8%), which well supports building it in the commodity pan-tilt camera chips.

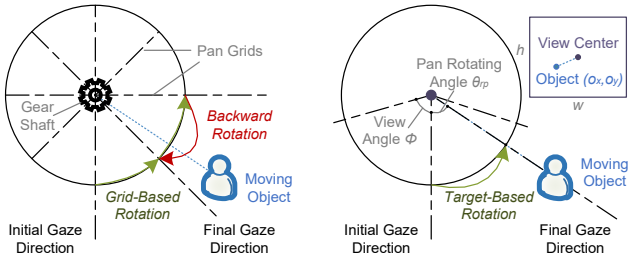
II. Motivation

This section introduces background of the state-of-the-art moving object tracking approaches of pan-tilt cameras, followed by real-world measurements of their performance which motivates our study.

A. Tracking with Pan-Tilt Cameras

Pan-tilt cameras are capable of automatic directional control that makes it possible to continuously track a moving object in wide areas with a single camera and adaptive to various surveillance environments. In spite of widespread adoption in indoor elderly/child caring and anti-theft alarming, commodity pan-tilt cameras have yet to be deployed in the energy-constrained outdoor environments[2]. To explore whether a pan-tilt camera can achieve long-term and energy-efficient moving object tracking, we first demystify the state-of-the-art tracking approaches adopted by two types of pan-tilt cameras.

[2]New outdoor wire-free and solar-powered cameras like Reolink Go [9] and Eufy SoloCam [10] have not provided pan-tilt rotations.



(a) Grid-Based Tracking (b) Target-Based Tracking

Fig. 1. Moving object tracking with two types of pan-tilt cameras (in the pan dimension).

Grid-Based Tracking. This tracking approach is typically applied by pan-tilt cameras with stepper motors. They divide a full rotation into equal steps and control the pan/tilt gear shaft to rotate one grid with a certain number of pulses [2]. On this basis, a camera conducts moving object detection in each video frame through vision detection algorithms like temporal frame differencing [11] or background subtraction [12]. Every time an object is detected, the camera seeks a pair of pan-tilt grids that makes the object closest to its gaze direction. Specifically, driven by the pan and tilt motors in sequence, the two gear shafts each rotate grid by grid towards the moving object and finally stop at a grid superior to the neighbor grids. When the target object is in the direction between two grids, the previous grid may make the object closer to the gaze direction compared with the current one, and this will trigger the gear shaft to rotate back to it (termed *backward rotation*). Fig. 1(a) gives a demonstration of the rotation process in the pan dimension.

Target-Based Tracking. In contrast to stepper motors, micro digital servo motors are controlled by a pulse of variable width that determines the pan/tilt rotating angle [3], which provides much higher rotation speed and position precision. The low cost, close to that of stepper motors (only ∼$1), makes it applicable to pan-tilt cameras. In light of the motors' properties, the most commonly-used basic tracking approach is to keep the camera targeting a detected object (the detection algorithm is similar to above). When the object center is not in the camera's view center, the pan-tilt gear shafts each rotate by a certain angle according to the proportion of pan/tilt view angle ϕ[3] to frame width w or height h. Given the object center's coordinates relative to the view center (o_x, o_y), the pan and tilt rotating angles are calculated respectively by

$$(\theta_{rp}, \theta_{rt}) = (-2o_x * \phi/w, 2o_y * \phi/h). \quad (1)$$

The pan-dimensional rotation in this approach is illustrated in Fig. 1(b), in comparison with the previous grid-based rotation.

In face of continuous violent rotations probably incurred by the above approach, an alternative approach is to set a fault-tolerant boundary around the view center. Provided that the target object keeps inside the boundary, the camera will not rotate as the object moves. The boundary width and height are

[3]The view angle ϕ is determined by the camera's focus length f and retina size d ($\phi = \arctan(d/2f)$).

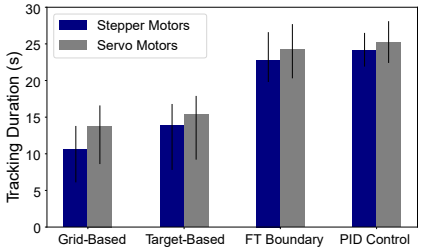


Fig. 2. Tracking duration of the state-of-the-art approaches on two types of pan-tilt cameras.

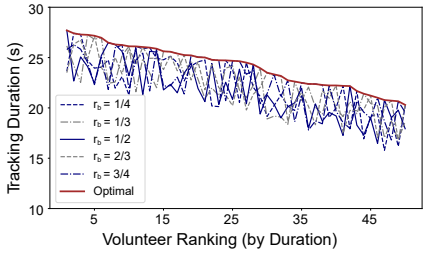


Fig. 3. Tracking durations achieved with different boundary-to-frame ratios.

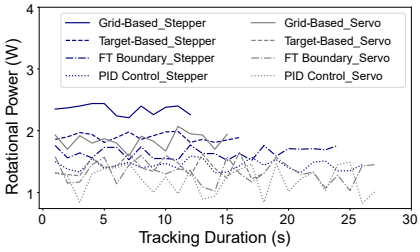


Fig. 4. Rotational power variances of the two cameras with different tracking approaches.

generally set to be in equal proportion to the camera's frame width and height, where the boundary-to-frame ratio is denoted by r_b. Another alternative approach is to adopt a controller employing feedback and correction based on proportional (P), integral (I), and derivative (D) terms [7]. It minimizes the error $e(t)$ between a measured process variable and the desired setpoint over time by adjusting a weighted sum of its PID,

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}, \qquad (2)$$

where K_p, K_i, and K_d are coefficients. In our case, we denote error values $e_h(t)$ and $e_v(t)$ as the horizontal and vertical distances between current object center and view center, and accordingly calculate $u_h(t)$ and $u_v(t)$ as the object's two-dimensional moving distances (towards the view center) with Eq. (2). On this basis, we can obtain the pan-tilt rotating angles based on Eq. (1) by replacing (o_x, o_y) with $(u_h(t), u_v(t))$.

B. Measurements and Observations

We next conduct measurements on the above-described state-of-the-art tracking approaches, carefully examining if their real-world performance meets the long-term and energy-efficient requirements of pan-tilt cameras' practical use.

Measurement Methodology. We mainly concern the following two metrics in our measurements:

- *Tracking duration* measures how long a pan-tilt camera can keep sight of an object by rotations. The metric is significant for surveillance, where it is desirable to record as many object moving details as possible for behavior analysis.
- *Rotational power consumption* measures the power increment incurred by pan/tilt rotations for object tracking. The metric is free from energy cost diversity among different rotations and tracking durations. It is of particular importance to those energy-constrained scenarios.

We study four state-of-the-art tracking approaches – grid-based tracking, basic target-based tracking and its two alternatives fault-tolerant (FT) boundary and PID control. We run each of them on pan-tilt cameras with stepper motors and servo motors, respectively[4]. In the stepper-driven camera, the angle between two adjacent grids (*i.e.,* a step) is 15°. We set the same virtual step for the servo-driven camera when conducting grid-based tracking on it. To grasp the stability of the performance, we recruit 50 volunteers with different body size and moving speed to conduct the tracking experiment in sequence.

Concretely, the two pan-tilt cameras are hung in the middle of the ceiling of a $\sim 15m^2$ room in turn, making the whole room in the coverage through gimbal rotations. Each volunteer walks around the room stochastically at his/her normal speed until the pan-tilt camera loses track of him/her, during which the movement may involve intermission and sudden changes in direction or speed. For the latter two tracking approaches, the volunteers conduct experiments with different values of parameter(s) r_b or (K_p, K_i, K_d), and we select the optimal ones that achieve longest tracking duration for the final result comparison. In the whole experiment, we connect a power meter to the pan-tilt cameras to measure their power consumption.

Tracking Duration Results. The average, maximum, and minimum tracking duration results of the 8 test items are depicted in Fig. 2, where each test item is denoted as *tracking approach_motor type*. As the figure shows, the tracking durations of all approaches are not ideally long (less than 30 seconds). Moreover, the durations of the stepper-driven camera are always shorter than the counterparts of the servo-driven camera. It is largely due to the slow start and stop property of the stepper motor, which makes the camera's rotation difficult to keep up with rapid object movements. Although servo motors and target-based tracking can help avoid inefficient operations, the basic target-based tracking approach is very sensitive to the object's small motions, which may lead to excessive mechanical rotations that are executed in an asynchronous manner.

In contrast, with properly selected parameters, the two alternative approaches can increase the tracking duration by reducing rotation intensity. However, the parameter(s) of both approaches may be affected by quite a few object-specific factors. As described in Fig. 3, the optimal (longest) tracking durations are achieved with quite different boundary-to-frame ratio r_b for volunteers with various body sizes, moving speeds and routes. In reality, it is difficult to set a proper-sized boundary for general object tracking in real time. If the boundary is not large enough, there may still be excessive rotations executed asynchronously; if it is too large, there remains little room for reaction to object moving out of the camera's field of

[4]The implementation is similar to that of WiseCam (illustrated in §IV-A).

TABLE I
AVERAGE ENERGY COST (WH) OF TWO CAMERAS WITH DIFFERENT TRACKING APPROACHES FOR ALL VOLUNTEERS.

Tracking Approaches	Stepper-Driven	Servo-Driven
Grid-Based Tracking	4.546	3.641
Target-Based Tracking	4.095	3.512
With FT Boundary	3.846	3.228
With PID Control	3.639	3.090

view. Likewise, the PID coefficients (K_p, K_i, K_d) have poor generalization to different motors and objects. In practice, the parameter tuning process is complex and costly, and improper values will result in frequent tracking failures.

Furthermore, repeatedly conducting object detection in each frame with the frame differencing algorithm also makes the camera prone to tracking failures. According to our measurements, a considerable proportion ($\sim$40%) of tracking failures occur when the volunteers move intermittently. This is because once a volunteer stops moving, the camera may take a pack of noise pixels in background as the next tracking target and thus lose sight of the volunteer. The drawback hinders the cameras from operating when there is more than one person, as the tracked target will frequently change among them and the camera cannot keep tracking anyone.

Power Consumption Results. We also monitor the two cameras' power variances with each state-of-the-art approach (adopting the optimal parameters) every second throughout each volunteer's tracking process. We can obtain the 8 test items' *incremental* rotational power consumption by deducting the two cameras' normal working power (on average 2.2 W and 1.5 W respectively) from the measured total power variances. We visualize the results of all test items for a volunteer who makes the longest total tracking duration with Fig. 4. Here we omit to show the other volunteers' power variances as they follow the same trend. As depicted in the figure, object tracking continuously brings considerable power consumption to both cameras. The output power of stepper motors is always high while that of servo motors is highly variable. By integral operation, we can further estimate the total energy cost of two cameras with the four approaches (taking their normal working power into account). As shown in Table I, the average energy cost results of 50 volunteers for all the test items are beyond 3 Wh. In view of the typical solar battery's energy budget (48 Wh/day at most) [13], the state-of-the-art tracking approaches would incur power cuts to the pan-tilt cameras in off-the-grid outdoor environments.

Opportunities. The real-world measurements indicate that the state-of-the-art tracking approaches are inadequate to provide long-term tracking and inapplicable to fulfill the energy-constrained scenarios. According to our observations, the root cause is the stateless nature of these approaches that determines gimbal rotations only based on instant object detection, increasing risk of tracking failure and energy costs. It is further due to the simplicity principle [8] that the above easy-to-implement tracking approaches are popular on today's pan-tilt cameras. Fortunately, we also notice opportunities to potentially address the unravelled obstacle. First, the pan-tilt camera should keep gazing at the detected object throughout the tracking process instead of detecting in each frame. Second, the pan-tilt camera's stay point could be decided based on the object's state variations to prevent dispensable rotations.

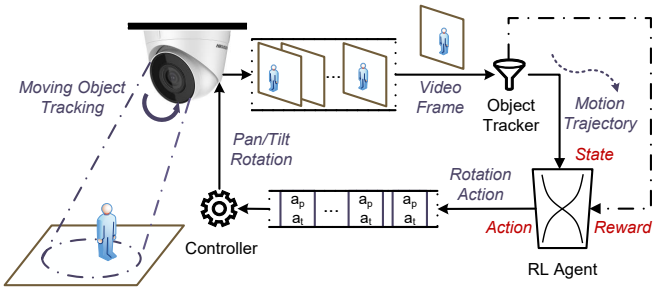


Fig. 5. An architectural overview of WiseCam.

III. WISECAM DESIGN

Guided by our observations, we design WiseCam, a cost-effective tracking approach of pan-tilt cameras that provides long-term moving object tracking with low rotational energy costs. We next describe our solution to achieve this goal.

A. System Overview

Fig. 5 depicts the high-level architecture of WiseCam. The pan-tilt camera's video frames are enqueued for moving object tracking. *Object Tracker* matches the same moving object that appears in different frames through correlation filtering on multiple features, and then constructs a motion trajectory in the camera's panoramic space with a sequence of critical points (§III-B). Taking as input the spatio-temporal object motion information, *RL Agent* continuously tunes amplitude of pan-tilt rotations with well-customized rewards. To accelerate model convergence for online determination on new objects, the agent fuses learning experience from the previous tracking (§III-C). The output rotation actions are cached in a queue and then executed to control the camera's pan-tilt rotations.

B. Long-Term Moving Object Tracking

We start with designing a scheme to track a moving object in a long term, which involves correlating the detected object across frames and constructing the object's motion trajectory.

Object Detection and Correlation. Most commodity pan-tilt cameras apply temporal frame differencing [11] as the object detection algorithm, which calculates the intersection of two pixelwise differences between consecutive video frames (F_{t-2}, F_{t-1}, F_t) to obtain a difference frame D_t (*i.e.,* $D_t = |F_t - F_{t-1}| \cap |F_{t-1} - F_{t-2}|$). After thresholding to avoid noise, the largest connected area in D_t is deemed as the target moving object. Despite its simplicity and low cost, the algorithm often results in incomplete object pixels and largely affects the positioning accuracy. To precisely obtain the object's complete contour, we ameliorate the existing detection

algorithm by integrating *regional* optical flow. Concretely, a frame is divided into a group of square regions, of which the side length is the greatest common divisor of frame width and height. For any frame F_t, we calculate a difference frame and extract the target object's pixels therein based on the above-described differencing algorithm. Then only for each pixel (x, y) in regions where the object is located (but not the whole frame), we estimate a two-dimensional optical flow vector $\vec{u} = (\mathrm{d}x/\mathrm{d}t, \mathrm{d}y/\mathrm{d}t)$ with the Lucas-Kanade (LK) algorithm [14], which confirms the object's contour and movement pattern between adjacent frames. By this means, we can acquire an object's complete contour in any frame and meanwhile reduce the computation overhead to the full.

To eliminate interference from background noise as well as other objects (as shown in §II-B), correlating the same object across frames is a high priority for long-term object tracking. Instead of reconducting object detection in each frame as in the state-of-the-art tracking approaches, here we leverage *correlation filtering* [15] to determine the target object in an arbitrary frame through element-wise operations in the frequency domain. Specifically, we obtain the object's updated position and size[5] in each frame with two filters. The position filter f_p conducts sampling on the frame image at twice object size and generates a set of sample images. Likewise, the scale filter f_s scales up and down the object size to form a small collection of candidates. We calculate the correlation of a filter $f_x(x = p, s)$ with each sample (image or size) h_k by

$$g_k = h_k \otimes f_x = \mathcal{F}^{-1}(\mathcal{F}(h_k) \odot \mathcal{F}(f_x)^*), \tag{3}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are 2-demensional *fast fourier transform* and its inverse operation, and $\odot$ and $*$ denote element-wise multiplication and complex conjugate, respectively. To locate the object in a frame, we find proper filters f_p and f_s as well as h_k's that achieve the largest correlation through a least-squares optimization method [15]. We recurrently execute the process until no object is correlated in three consecutive frames, in which case we re-detect a moving object. Note that such a scheme enables the target moving object to be kept tracking even when there are multiple objects visible to the camera.

Motion Trajectory Construction. Through the above efforts, we can continuously acquire worthy object information like position, size, and accurate contour. We next construct a trajectory for the target object with such information, which is conducive to wisely determining the camera's rotations. To reduce the influence of small motions, we represent the object's position by its centroid instead of the center of its circumscribed square. Given a set of pixels on and inside the contour $P = \{(p_x^i, p_y^i)\}$ $(i = 1, \cdots, n)$, we calculate the object centroid's coordinates (c_x, c_y) by $c_v = [(\sum_{i=1}^{n} p_v^i)/n]$ $(v = x, y)$[6]. To keep the object staying in the camera's field of view, we also concern position variations of the object's contour points that are closest to the frame image's boundaries.

[5]Here the two factors are reflected by two-dimensional coordinates of object center and side length of the object's circumscribed square, respectively.

[6]Here the operator [] means rounding the number to the nearest integer.

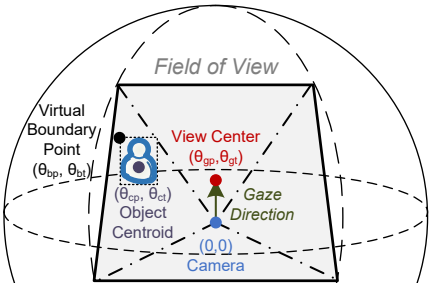


Fig. 6. Coordinates in the panoramic space.

Accordingly, we customize a virtual boundary point (b_x, b_y) by abstracting x-y coordinates from the contour points that respectively maximize horizontal and vertical distances to the image center, *i.e.,* $b_v = \max_i\{|p_v^i|\}$ $(v = x, y)$. We term the virtual boundary point together with object centroid as *critical points*.

Given that centers of different frames may be on various angles as the camera rotates, we need further transform the above-calculated coordinates relative to a frame's image center to absolute coordinates in a unified space. As depicted in Fig. 6, a 360-degree panoramic space is generated by the camera's pan-tilt rotations. The camera's gaze direction can be represented by pan-tilt rotating angles $(\theta_{gp}, \theta_{gt})$ (ranging from $[-1/2\Theta_u, 1/2\Theta_u]$ $(u = p, t)$, where Θ_p and Θ_t are maximum rotating angles provided by the camera manufacturer). We notice that the relative x-y coordinates of an object's critical points in a frame are proportional to pan-tilt angle differences between the real object and view center, for which their pan-tilt rotating angles relative to the camera's view center can be obtained based on Eq. (1). Therefore, we calculate the critical points' absolute coordinates in the panoramic space by

$$(\theta_{vp}, \theta_{vt}) = (\theta_{gp} - v_x * \phi/w, \theta_{gt} + v_y * \phi/h), \tag{4}$$

where $v = c, b$ represents the two critical points, w and h are frame width and height, and ϕ is the camera's view angle. Finally, an object's motion trajectory in frame F_m can be expressed by a sequence of quadruples that represent the object's critical points in previous frames,

$$\vec{T_o} = \{(\theta_{cp}^i, \theta_{ct}^i \theta_{bp}^i, \theta_{bt}^i)\}, \quad i = 1, \cdots, m. \tag{5}$$

C. Online Rotation Determination

The above-described object tracking scheme enables the camera to obtain the target object's up-to-date motion trajectory in its panoramic space in real time, which indicates the object's previous state variations in multiple dimensions (*e.g.,* position, size, speed, and direction) and can be leveraged to deduce the object's next state. We next design a reinforcement learning (RL) model to conduct appropriate pan-tilt rotations at runtime with the motion trajectory information.

RL Model Formulation. An RL agent is built to keep observing the target object's instantaneous state S_i in any frame F_i and to determinate the pan-tilt rotation action a_i, which is expected to minimize the camera's costs while

maintaining object tracking. Then the rotation action a_i will be executed to reorient the camera and produce a new state of the object S_{i+1} that initiates a new round of learning and determination. The mapping of $S_i \rightarrow a_i$ is determined by the RL model's control policy $\pi(S_i, a_i)$, which is learnt and updated during iterative state observation and action execution.

As the input of the RL agent, the state can be assembled with the camera's present gaze direction $(\theta_{gp}, \theta_{gt})$ and the latest k quadruples of the object's motion trajectory $\vec{T}_o$. According to Eq. (5), the state in the current frame F_m is denoted with

$$S_m = (\vec{\theta}^m_{cp}, \vec{\theta}^m_{ct}, \vec{\theta}^m_{bp}, \vec{\theta}^m_{bt}, \theta^m_{gp}, \theta^m_{gt}), \tag{6}$$

where the first four elements are sequences extracted from the latest k quadruples by column, *e.g.,* $\vec{\theta}^m_{cp} = \{\theta^i_{cp}\}$ $(i = m-k+1, \cdots, m)$. For a given state S_m, the RL agent designates the pan-tilt rotation action $a_m = (a^m_p, a^m_t)$. In order to accommodate online determination, a^m_p and a^m_t are both selected from a compact discrete action space $\mathcal{A} = \{\omega j\}$ $(j \in [-n_{au}, n_{au}] \cap Z; \omega n_{au} < 1/2\Theta_u, u = p, t)$, where ω is the angle of a rotation unit, and (n_{ap}, n_{at}) are rotation amplitudes for two dimensions.

To make the RL agent learn from past experience, each action is associated with a reward, which takes both tracking status and rotation cost into account. The reward r_m of action a_m is calculated when the next state S_{m+1} is produced. At this time, we can exactly acquire the target object's new position and accordingly update its motion trajectory. When the object gets out of the camera's field of view, the camera rotates immediately to pan-tilt angles of the latest object centroid, which generally resumes tracking of the object at a marginal cost. In this case, we set *loss reward* r^m_l as an enough large negative value (*e.g.,* -10) to embody the last action's extremely adverse effect. If the object is still in the camera's field of view, we concern its distances to the view boundaries as well as moving direction. Given the camera's new gaze direction $(\theta^{m+1}_{gp}, \theta^{m+1}_{gt})$ and the updated object motion sequences $(\vec{\theta}^{m+1}_{cp}, \vec{\theta}^{m+1}_{ct}, \vec{\theta}^{m+1}_{bp}, \vec{\theta}^{m+1}_{bt})$ in the state S_{m+1}, we calculate *position reward* r^m_p and *direction reward* r^m_d respectively by

$$r^m_p = 1 - 2|\theta^{m+1}_{bp}|/\Theta_p - 2|\theta^{m+1}_{bt}|/\Theta_t, \tag{7}$$

$$r^m_d = -\frac{\theta^{m+1}_{cp}\theta^{\Delta}_{cp} + \theta^{m+1}_{ct}\theta^{\Delta}_{ct}}{(|\theta^{m+1}_{cp}| + |\theta^{m+1}_{ct}|)(|\theta^{\Delta}_{cp}| + |\theta^{\Delta}_{ct}|)}, \tag{8}$$

where $\Delta\theta_{cu} = \theta^{m+1}_{cu} - \theta^m_{cu}$ $(u = p, t)$. In addition, we adopt *cost reward* r^m_c to embody costs of pan-tilt rotations,

$$r^m_c = 1 - a^m_p/(\omega n_{ap}) - a^m_t/(\omega n_{at}). \tag{9}$$

To penalize loss of tracking and high rotation cost, the overall reward r_m should comprise all the above-described reward metrics, and we define it as

$$r_m = \mu r^m_l + (1-\mu)(\beta r^m_p + \delta r^m_d + \eta r^m_c), \tag{10}$$

where $\mu \in \{0, 1\}$ indicates if the action makes the object get rid of the camera's field of view ($\mu = 1$ means that loss of tracking occurs), and β, δ, and η are all hyper-parameters to be adjusted.

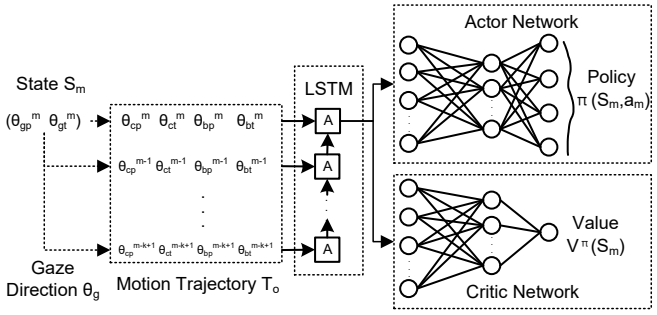


Fig. 7. The RL model of WiseCam.

The RL model's internal structure is illustrated in Fig. 7. Each input state S_m is transformed into a combination of four sequences $(\vec{\psi}^m_{cp}, \vec{\psi}^m_{ct}, \vec{\psi}^m_{bp}, \vec{\psi}^m_{bt})$ by $\vec{\psi}^m_{vu} = \{\theta^i_{vu} - \theta^i_{gu}\}$ $(i = m-k+1, \cdots, m; v = c, b; u = p, t)$. To reason implicit spatio-temporal features hidden in the time series, the four sequences are fed into a long-short-term-memory (LSTM) structure [16] with k cells. The sequences flattened by LSTM cells are transmitted into two similar fully-connected neural networks (NNs) – one outputs a rotation action based on features extracted (termed *actor*), and the other one is served as a *critic* that judges the action's value. Each NN contains two hidden layers and an output layer. The hidden layers employ the *tanh* activation function [17] to enhance learning capabilities. The output layer of the actor network takes *softmax* [18] as the activation function that generates probability distribution of each candidate action and finally provides the action with largest probability, while that of the critic network estimates the expected total reward starting at the present state.

Fast-Convergent Training. To adapt with the requirement of online determination, the RL model should converge (*i.e.,* stably generate reasonable decision) as soon as possible. Here the training objective is to maximize the total cumulative reward that the model receives. Policy gradient [19] is an effective approach to achieving this goal, but it may suffer from the problem of difficult convergence caused by substantial policy changes. In view of this, we train the model with a fast-convergent algorithm named proximal policy optimization (PPO) [20] to avoid updates of parameter ξ that change the control policy $\pi_\xi(S_m, a_m)$ too much at each step. Specifically, we formulate the gradient of cumulative discounted reward with an *advantage function* $A_\xi(S_i, a_i)$[7]. In practice, each update of ξ follows the policy gradient,

$$\xi \leftarrow \xi + \alpha \sum_i \nabla_\xi \log \pi_\xi(S_i, a_i) A_\xi(S_i, a_i), \tag{11}$$

where α is the learning rate. For any given (S_i, a_i, r_i, S_{i+1}), $A_\xi(S_i, a_i)$ can be estimated by $r_i + \gamma v_\xi(S_{i+1}) - v_\xi(S_i)$, where γ serves as a discounting factor. To help the RL model fast

[7] $A_\xi(S_i, a_i)$ represents difference in the expected reward for action a_i in state S_i, compared with averaged reward for actions drawn from policy π_ξ.

converge to a good policy, we bound the difference between new and old policies and adopt their probability ratio $r_\xi = \pi_\xi(S_i, a_i)/\pi'_\xi(S_i, a_i)$ to replace $\pi_\xi(S_i, a_i)$ in Eq. (11). On this basis, the model's loss function is devised as

$$L(\xi) = E(\min\{r_\xi A_\xi, c(r_\xi, 1-\epsilon, 1+\epsilon)A_\xi\}), \qquad (12)$$

where the ratio r_ξ is clipped to be within the small interval $[1-\epsilon, 1+\epsilon]$ by function $c()$, and A_ξ is short for the above $A_\xi(S_i, a_i)$. By this means, the model will update policy parameters smoothly and avoid jumpy decisions, so as to achieve fast convergence.

Multi-Model Aggregation. With the above fast-convergent training algorithm, the RL agent still needs to learn a certain amount of object motion data before making a wise rotation determination. According to our observation, after entering the camera's field of view, an object generally stays for a period of time that requires no camera rotations (termed *quiescent period*), which facilitates initial training with object motion data. However, only based on data collection in the quiescent period, the model sometimes makes unreasonable rotation determinations for the subsequent object tracking, *e.g.,* rotating dramatically for objects close to the view center, or keeping still when an object tends to get rid of the field of view.

To overcome this issue, we make the RL agent draw on learning experience from previous object tracking of the same type. It has been illustrated that averaging the parameters of the same neural cell across a number of NN models can achieve aggregating these models' experience [21]. We thus conduct multi-model aggregation by fusing the NN parameters of previous objects' models. Suppose there are totally n_m models of objects with the same structure, and each model contains n_c cells. The NN parameters are denoted with a matrix $\Xi_{ij} = (\xi_{ij})$, of which ξ_{ij} indicates the parameter of j-th cell in the i-th model. We perform a weighted averaging operation following the principle of federated learning [21] to calculate each element in the aggregated model, $\bar{\xi}_{ij} = \sum_{i=1}^{n_m} \rho_i \xi_{ij}$, where ρ_i represents the experience weight of model i. We generally set equal weights for all previous objects. Once an object has been tracked before (judged by a ReID approach [22]), we set a prioritized weight as it provides more experience. Through multi-model aggregation, the RL agent can accelerate NN generalization during the camera's quiescent period.

IV. Evaluation

In this section, we first briefly present implementation of WiseCam, and then demonstrate its tracking cost-effectiveness with real-world human tracking experiments as well as data-driven testing of large-scale human trajectories.

A. System Implementation

To embody WiseCam's generality for moving object tracking, we implement it on the two types of pan-tilt cameras introduced in §II-A. Note that the stepper-driven commodity pan-tilt cameras do not support directly running processes on their platforms. Instead, most of them provide public interfaces for status acquisition and rotation execution based on a common protocol ONVIF [23]. By contrast, the servo-driven pan-tilt cameras should generally be built by oneself with a small tripod head and a fixed camera, and controlled by PWM drivers [24]. In view of the two heterogeneous cameras' properties, we build a prototype of WiseCam on Raspberry Pi 4B [25] that communicates with them through WiFi and I2C bus, respectively. On this basis, the prototype controls gimbal rotations of the motors by invoking ONVIF or PWM APIs.

We implement both components of WiseCam in Python. *Object Tracker* utilizes the OpenCV library [26] for frame image processing, which helps achieve object detection and cross-frame correlation. To reduce the computation overhead, we set frame size to be the camera settings' minimum value 640×480 pixels. *RL Agent* builds the NN models with TensorFlow [27]. In each state we input the latest $k = 10$ quadruples of the motion trajectory into the model's LSTM layer. Based on *grid search* [28], we set the unit number of FC layer as 32, set the learning rate α and discounting factor γ as 0.1 and 0.95, and empirically set the weights of rewards β, δ, and η to be 1, 1, and 2, respectively. We employ Adam optimizer [29] to update the gradient descent.

B. Experiment Setup

We take a Raspberry Pi 4B equipped with a 4-core CPU @1.5GHz and 2 GB of memory as the deployment platform. It communicates with a typical commodity pan-tilt camera with stepper motors through WiFi with 100 Mbps of maximum bandwidth, and controls a self-built pan-tilt camera with servo motors by I2C bus at a transfer rate of 3.4 Mbps. The two cameras' maximum pan-tilt rotating angles (Θ_p, Θ_t) and view angle ϕ are (330°, 150°), 115° and (210°, 130°), 120°, respectively. Besides, we connect each camera with a power meter to measure power consumption.

Metrics and Baselines. In view of the pan-tilt cameras' practical requirements, we concern two metrics – tracking duration and rotational power consumption, which are identical to those in §II-B. Among those measurement results of the state-of-the-art tracking approaches, both fault-tolerant boundary and PID control mechanisms can bring better performance to the target-based approach. Therefore, we simultaneously integrate them into the basic target-based approach (with optimal parameters preset) and adopt the improved approach as a baseline for comparison. We also take the grid-based approach that is widely adopted by commodity cameras as another baseline in regardless of its performance. We compare WiseCam against the two baselines on both types of pan-tilt cameras.

Testing Datasets. Similar to measurements in §II-B, we track the 50 recruited volunteers' motions when they walk around a room stochastically with two pan-tilt cameras controlled by WiseCam and the two baselines. In addition, to evaluate WiseCam's tracking performance in the outdoor environment, a large-scale data trace of human trajectories is crowdsensed from 1052 students who walk daily with mobile devices over a semester, during which geolocations are collected every 15

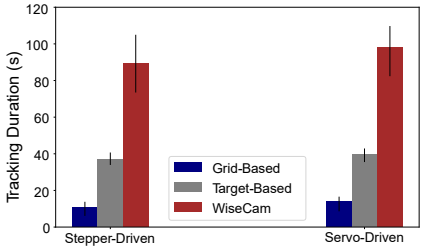


Fig. 8. Tracking Duration of WiseCam compared with baselines on two types of pan-tilt cameras.

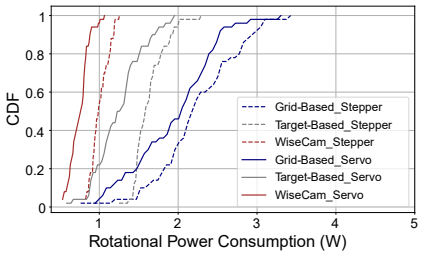


Fig. 9. CDFs of rotational power consumption by tracking all volunteers.

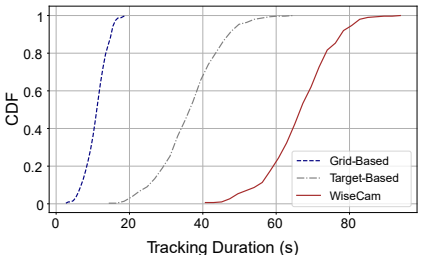


Fig. 10. CDFs of tracking durations conducted with large-scale trajectory data.

seconds (over 60 million samples in total). According to the monitoring range of typical outdoor cameras, we cluster the samples and extract trajectory segments in 10 circle areas with a radius of 20 meters, where the samples are highly concentrated (3024 segments in total). Then we virtually put a stepper-driven pan-tilt camera in the center of each area (the view angle should be 40° for the preset range), and accordingly transform the collected geolocations into coordinates in the camera's panoramic space. Based on the transformed trajectories with 15-second intervals, we can emulate moving object tracking (only conducting the pan-tilt rotation determination) with WiseCam and the two baselines. In this case, the object is abstracted into a point, and coordinates of its centroid and virtual boundary point are the same.

C. End-to-End Performance

We start with measuring WiseCam's overall performance. Note that "Target-Based" in all the following experimental figures refers to the *improved* target-based tracking approach.

Improvement at Tracking Duration. We first evaluate how long WiseCam can keep sight of the 50 volunteers in the real-world tracking experiment. To quantify this, we show in Fig. 8 WiseCam's average, maximum and minimum tracking durations for the 50 tracking tests with both types of cameras in contrast with those of two baselines. We can see from the figure that WiseCam keeps tracking the target object for overall 2.4 to 8.4 times as long as the baselines. The improvement mainly comes from WiseCam's cross-frame object correlation and wise rotation determination, which helps overcome the drawbacks of baselines illustrated in §II-B.

Reduction at Power Consumption. We also concern energy cost incurred by camera rotations during the above experiment. Accordingly, we monitor the two involved cameras' power variances every 20 seconds. We average the power monitoring values in each tracking duration to eliminate the influence of test items' tracking duration difference. The cumulative distribution functions (CDFs) of the six test items' rotational power consumption for the 50 tracking tests are plotted in Fig. 9. The two baselines both exhibit an apparently longer tail, and the reduction by WiseCam is 48.9% on average and at least 38.1%. It is largely due to WiseCam's ability to refrain from a multitude of dispensable gimbal rotations that bring remarkable energy cost. According to the result comparison given in Table II, WiseCam can reduce the servo-driven camera's energy cost to ∼2 Wh. It can well support daily object tracking of pan-tilt cameras powered by a typical solar battery with the energy budget 48 Wh/day [13].

TABLE II
AVERAGE ENERGY COST (WH) OF TWO CAMERAS FOR ALL VOLUNTEERS WITH WISECAM AND TWO BASELINES.

Tracking Approaches	Stepper-Driven	Servo-Driven
Grid-Based Tracking	4.546	3.641
Target-Based Tracking (Imp.)	3.535	2.894
WiseCam	3.056	1.978

Performance on Large-Scale Data. We further conduct large-scale object tracking emulation with WiseCam and the two baselines based on the transformed human trajectories (*cf.* §IV-B). Through comprehensive testing of tracking all 3024 trajectory segments on the configured virtual camera, we describe the CDF of tracking duration for the three approaches in Fig. 10. Similar to results of the real-world human tracking experiment, both the median and tail durations of WiseCam (62.1 s and 88.4 s) are much longer than those of the baselines, which suggests WiseCam's high effectiveness given the large total number of trajectories.

D. Micro Benchmarks

We next study the impact of individual components inside WiseCam (shown in Fig. 5) by comparing them with some alternative methods acting on the human tracking traces.

Tracking Accuracy vs. Efficiency. We contrast our moving object tracking scheme with three alternative approaches – repeatedly detecting objects with temporal frame differencing (TFD) [11], optical flow (OF) [14], and Tiny YOLOv3 [30]. We rerun the four algorithms on the videos recorded during the previous human tracking experiment. We manually label the target object's position in each video frame as the ground truth. Given that today's pan-tilt cameras are not equipped with GPU, here we execute all the algorithms only by CPU. Fig. 11 shows their (a) tracking failure rate and (b) per-frame execution time. Although YOLOv3 provides the lowest failure rate, its execution speed is slowest among the algorithms[8]. In

[8]YOLO's inference time is ∼50 ms when executed with well-performed GPUs [30], but an embedded processor will slow it down by over 10 times.

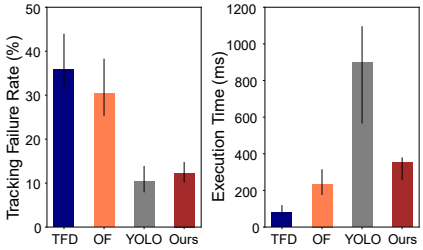


Fig. 11. Failure rate and per-frame execution time comparisons among tracking algorithms.

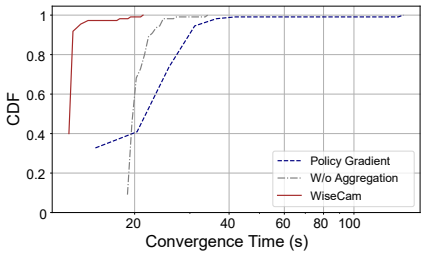


Fig. 12. The influence of training algorithm and aggregation mechanism to the convergence time.

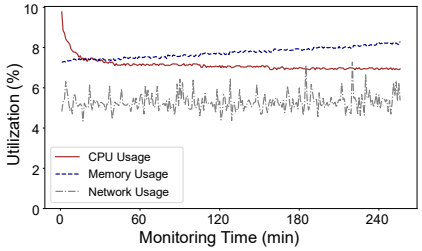


Fig. 13. WiseCam's resource usages during the moving object tracking process.

contrast, our algorithm achieves a good balance between accuracy and efficiency, which fits tracking with pan-tilt cameras.

Convergence of RL Model. We further concern how fast the RL model can converge and also the effects of training algorithm and model aggregation mechanism on this. Accordingly, we take adopting the aforementioned policy gradient algorithm [19] for training ("Policy Gradient") and adopting the PPO algorithm for training without model aggregation ("W/o Aggregation") as two comparison methods. Here we consider 10 successive determinations that keep the object in the camera's field of view as a signal of model convergence. Based on this notion, we observe convergence time of WiseCam and two comparison methods on tracking the 50 volunteers, and plot their CDFs in Fig. 12. The comparisons validate the PPO algorithm's fast convergence in practice and demonstrate effectiveness of model aggregation.

Resource Usages. We finally monitor the platform Raspberry Pi's resource usages during the stepper-driven pan-tilt camera's human tracking experiment. In addition to the basic CPU and memory variations, we also measure its network usages given that rotation actions are transferred to the camera through WiFi. As shown in Fig. 13, the usages of CPU, memory, and network (with combined inbound and outbound traffic) are quite low and steady (all in the range of 5% to 8%) confronted with intensive rotations, which demonstrates that the resource of a single Raspberry Pi is more than enough to handle the pan-tilt camera's moving object tracking.

V. Related Work

We discuss closely related work in the following two areas.

Camera Video Analytics. With the prevalence of IoT cameras in today's society, there has been a quantity of work on video analytics for surveillance. Most recent studies process live videos streamed from the camera on edge or cloud servers with NN-based object detection models. Specifically, Chameleon [31] and VideoStorm [32] dynamically pick the best NN configuration to balance accuracy and resource overhead. Focus [33] and NoScope [34] improve the efficiency of video analytics pipelines by filtering out frames that lack relevant information for query. In addition, Reducto [35] achieves on-camera filtering by adapting filtering decisions, and Elf [13] executes video object counting under energy constraint. In contrast, our work conducts video analytics towards more rigorous requirements – serving real-time rotations with lowest possible energy cost.

Pan-Tilt Camera Tracking. Different from wireless signal-based tracking methods facilitated by sensing devices [36], [37], pan-tilt cameras adopt visual tracking together with two-dimensional rotations to keep sight of moving objects. As a practical functionality, moving object tracking has received tremendous attention in the computer vision field. Today's commodity pan-tilt cameras repeatedly conduct object detection with intuitive algorithms [11], [12]. In spite of its low cost, this approach often results in low accuracy of object positioning. Another popular tracking approach is to recognize objects in each frame with a CNN classification model (*e.g.,* YOLO [38], R-CNN [39]), which however can hardly guarantee tracking timeliness due to the long execution time. Moreover, some studies apply the above algorithms in pan-tilt camera tracking [4], [40], [41], but they all adopt the aforementioned inefficient target-based tracking approach. Our work overcomes the drawbacks by designing a correlation-filtering-based tracking scheme and an RL-based rotation determination model, which help achieve cost-effective moving object tracking.

VI. Conclusion

We present WiseCam, a cost-effective moving object tracking solution for pan-tilt cameras in this paper. We first reveal the ineffectiveness and high cost of the state-of-the-art tracking approaches by real-world measurements. Guided by our observations from measurements, we design enabling mechanisms of long-term moving object tracking and online rotation determination. We put the techniques together and implement an open-source WiseCam prototype, and then show its high tracking duration and low power consumption with comprehensive evaluations. In practice, WiseCam can benefit pan-tilt cameras in diverse scenarios, especially when the cameras' energy is constrained.

Acknowledgment

This research is supported in part by the National Key R&D Program of China under grant 2021YFB2900100, the National Natural Science Foundation of China under grant 62102224, the Beijing Natural Science Foundation under grant 4222026, and the Engineering Research Center of Ministry of Education on Database and Business Intelligence.

REFERENCES

[1] Q. Research, "Global PTZ Camera Market Research Report 2020," *Tech Report MSR2422104*, pp. 1–120, 2020.

[2] Wikipedia, "Stepper Motor," 2022, https://en.wikipedia.org/wiki/Stepper_motor.

[3] I. Products, "Stepper Motor Fundamentals," 2022, https://islproducts.com/design-note/servo-motor-fundamentals.

[4] K. Okumura, H. Oku, and M. Ishikawa, "High-speed Gaze Controller for Millisecond-order Pan/tilt Camera," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 6186–6191.

[5] Z. Wu and R. J. Radke, "Keeping a Pan-Tilt-Zoom Camera Calibrated," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 35, no. 8, pp. 1994–2007, 2013.

[6] M. Rovai, "Automatic Vision Object Tracking," 2018, https://towardsdatascience.com/automatic-vision-object-tracking-347af1cc8a3b?gi=661a553688.

[7] Wikipedia, "PID Controller," 2022, https://en.wikipedia.org/wiki/PID_controller.

[8] ChangingMinds, "Simplicity Principle," 2022, http://changingminds.org/principles/simplicity.htm.

[9] ReoLink, "ReoLink Go: Wire-Free Security Goes Anywhere with 4G LTE," 2022, https://reolink.com/product/reolink-go.

[10] Eufy, "Pre-Order SoloCam Series," 2022, https://us.eufylife.com/pages/solocam-preorder.

[11] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving Target Classification and Tracking from Real-Time Video," in *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1998, pp. 8–14.

[12] M. Piccardi, "Background Subtraction Techniques: A Review," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2004, pp. 3099–3104.

[13] M. Xu, X. Zhang, Y. Liu, G. Huang, X. Liu, and F. X. Lin, "Approximate Query Service on Autonomous IoT Camera," in *Proceedings of 18th ACM International Conference on Mobile Systems, Applications and Services (MobiSys)*. ACM, 2020, pp. 191–205.

[14] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of Imaging Understanding Workshop*. DARPA, 1981, pp. 121–130.

[15] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual Object Tracking using Adaptive Correlation Filters," in *Proceedings of Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 2544–2550.

[16] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] Y. L. Cun, I. Kanter, and S. A. Solla, "Eigenvalues of Covariance Matrices: Application to Neural-Network Learning," *Physical Review Letters*, vol. 66, no. 18, pp. 2396–2399, 1991.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006, p. 236.

[19] M. Pirotta, M. Restelli, and L. Bascetta, "Adaptive Step-Size for Policy Gradient Methods," in *Proceedings of Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2013, p. 1394–1402.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," in *arXiv preprint*, 2017, p. arXiv:1707.06347.

[21] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*. AISTATS, 2017, pp. PMLR 54:1273–1282.

[22] A. Bedagkar-Gala and S. K. Shah, "A Survey of Approaches and Trends in Person Re-Identification," *Image and Vision Computing*, vol. 32, no. 4, pp. 270–286, 2014.

[23] ONVIF, "ONVIF Profile S Specification v1.3," 2019.

[24] Adafruit, "Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface - PCA9685," 2021, https://www.adafruit.com/product/815.

[25] R. Pi, "Raspberry Pi 4: Your Tiny, Dual-Display, Desktop Computer," 2021, https://https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

[26] OpenCV, "OpenCV 4.5.0," 2020, https://opencv.org/opencv-4-5-0/.

[27] Google, "TensorFlow: An End-to-End Open Source Machine Learning Platform," 2021, https://tensorflow.google.com/.

[28] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the Relationship between Classical Grid Search and Probabilistic Roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[29] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *arXiv preprint*, 2014, p. arXiv:1412.6980.

[30] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," in *arXiv preprint*, 2018, p. arXiv:1804.02767.

[31] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable Adaptation of Video Analytics," in *Proceedings of Annual Conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM)*. ACM, 2018, p. 253–266.

[32] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live Video Analytics at Scale with Approximation and Delay-tolerance," in *Proceedings of 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, 2017, p. 377–392.

[33] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu, "Focus: Querying Large Video Datasets with Low Latency and Low Cost," in *Proceedings of 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX, 2018, p. 269–286.

[34] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "NoScope: Optimizing Neural Network Queries over Video at Scale," in *Proceedings of the VLDB Endowment*, vol. 10, no. 11. VLDB, 2018, pp. 1586–1597.

[35] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics," in *Proceedings of Annual Conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM)*. ACM, 2020, p. 359–376.

[36] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," in *Proceedings of the First IEEE International Conference in Pervasive Computing and Communications (PerCom)*. IEEE, 2003, pp. 1–9.

[37] K. Qian, C. Wu, Z. Yang, Y. Liu, and K. Jamieson, "Widar: Decimeter-Level Passive Tracking via Velocity Monitoring with Commodity Wi-Fi," in *Proceedings of the Eighteenth International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. ACM, 2017, pp. 6:1–10.

[38] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 6517–6525.

[39] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 580–587.

[40] D. D. Doyle, A. L. Jennings, and J. T. Black, "Optical Flow Background Estimation for Real-Time Pan/Tilt Camera Object Tracking," *Measurement*, vol. 48, pp. 195–207, 2014.

[41] S. Hu, K. Shimasaki, M. Jiang, T. Senoo, and I. Ishii, "A Simultaneous Multi-Object Zooming System Using an Ultrafast Pan-Tilt Camera," *IEEE Sensors Journal*, vol. 21, no. 7, pp. 9436–9448, 2021.