Multimed Tools Appl (2016) 75:17393–17419
DOI 10.1007/s11042-015-3003-9

Dynamic node selection in camera networks based on approximate reinforcement learning

Qian Li[1,2] · **Zhengxing Sun**[1] · **Songle Chen**[1] · **Shiming Xia**[2]

Received: 30 January 2015 / Revised: 27 August 2015 / Accepted: 9 October 2015 /
Published online: 7 November 2015

Abstract In camera networks, dynamic node selection is an effective technique that enables video stream transmission with constrained network bandwidth, more economical node cooperation for nodes with constrained power supplies, and optimal use of a limited number of display terminals, particularly for applications that need to obtain high-quality video of specific targets. However, the nearest camera in a network cannot be identified by directional measurements alone. Furthermore, errors are introduced into computer vision algorithms by complex background, illumination, and other factors, causing unstable and jittery processing results. Consequently, in selecting camera network nodes, two issues must be addressed: First, a dynamic selection mechanism that can choose the most appropriate node is needed. Second, metrics to evaluate the visual information in a video stream must be modeled and adapted to various camera parameters, backgrounds, and scenes. This paper proposes a node selection method based on approximate reinforcement learning in which nodes are selected to obtain the maximum expected reward using approximate Q-learning. The Q-function is approximated by a Gaussian Mixture Model with parameters that are sequentially updated by a mini-batch stepwise Expectation–Maximization algorithm. To determine the most informative camera node dynamically, the immediate reward in Q-learning integrates the visibility, orientation, and image clarity of the object in view. Experimental results show that the proposed visual

✉ Zhengxing Sun
szx@nju.edu.cn; http://cs.nju.edu.cn/szx/

Qian Li
public_liqian@163.com

Songle Chen
chensongle@hotmail.com

Shiming Xia
15295753714@163.com

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China

[2] College of Meteorology and Oceanography, PLA University of Science and Technology, Nanjing 211101, China

evaluation metrics can effectively capture the motion state of objects, and that the selection method reduces camera switching and related errors compared with state-of-the art methods.

Keywords Camera selection · Approximate reinforcement learning · Gaussian mixture model (GMM) · Video analysis · Camera networks

1 Introduction

Camera networks have recently received increased research interest for the effective resolution of partial and full occlusions, and the continuous tracking of targets over large areas where the limited field of view (FOV) of a single camera is insufficient. Such networks are used extensively in areas such as security, surveillance, human–computer interaction, navigation, and positioning. However, the utilization of camera networks is beset by challenges related to the deployment and control of the camera nodes, real-time fusion of high-resolution and high frame-rate video streams, and selection and coordination of the cameras [5, 30]. Camera selection involves determining which one (or more) camera(s) should be selected to obtain the maximum information or comfortable visual effects. Thus, camera selection is a fundamental challenge in applications such as target tracking and positioning [10, 18], area coverage [29], surveillance and behavior analysis [24, 33], and video summarization [8]. Dynamic node selection is an effective solution to problems such as video stream transmission with constrained network bandwidth, more economical node cooperation for nodes with constrained power supplies, and a limited number of display terminals. Such issues are faced by emerging applications such as security surveillance and smart homes, which need to obtain high-quality video of specific targets. For example, from the cameras deployed to survey some critical area, users need select only one optimal viewpoint to be displayed on a mobile device. However, in contrast to node selection in general sensor networks [11, 21], the nearest camera in a camera network cannot be identified by directional measurements alone. Furthermore, some errors are introduced in computer vision algorithms by complex background, illumination, and other factors, which cause processing results to be unstable and jittery in the time sequence. Consequently, in selecting nodes in camera networks, two issues must be addressed: First, a dynamic selection mechanism should be established that ensures the selected node captures the maximum expected amount of information and outputs comfortable video based on current and historical observations. Second, metrics to evaluate the visual information of a video stream must be modeled according to the demands of applications, and these should be adapted to various camera parameters, background, and scenes.

Previous studies have investigated the issue of node selection in camera networks from the viewpoint of selection mechanisms and visual information evaluation. The consequent methods proposed for node selection in camera networks can be divided into three categories: In the first category, the node that maximizes the current visual information is selected. For example, Tessens et al. [33] proposed a distributed principle viewpoint determination method for smart cameras, in which factors such as the visibility, direction of movement of targets, distance between the center of mass of the object and the camera, and the results of face detection are summarized into a score. The camera with the maximum score is then designated as the principle viewpoint. In [23], cameras are ranked and selected according to the target's location and look-up tables stored locally at each node. A camera selection approach for object tracking was presented by Monari et al. [22], in which the relevant subset of cameras is determined according to the current or predicted state of the target object and prior geometrical knowledge.

However, most methods in this category do not consider the dynamics of visual information and a target's state in scenes, overall visual effects, network communications, or energy costs. The second category of node selection methods uses a reward or cost function to integrate the constraints, and the selection process is optimized using historic and current system states. For example, in a method proposed by Li and Bhanu [16], the best-view estimation is obtained by iterative bargaining based on game theory. A best-view selection algorithm that generates a smooth video sequence was proposed by Jiang et al. [13], who solved the resulting optimization problem using dynamic programming. Daniyal et al. [7] presented a best-view selection method based on a dynamic Bayesian network according to the states taken from a certain historical time window, and used the resulting output for video production. Although these methods do not predict the state transitions and the relay of rewards, the fact that the reward will be reflected after the action, and thus affect subsequent states and actions, is not considered. In the third category, dynamic selection decisions are made according to current system states and predicted future states. For example, Partially Observable Markov Decision Processing (POMDP) may be used to find the optimal scheduling policy in tracking and surveillance camera networks [17, 31] by predicting the future reward of an action according to certain assumptions. However, these methods require clear state transitions, which are difficult to explicitly define in complex dynamic systems. Additionally, the computational complexity grows exponentially with the state space because of the "curse of dimensionality."

On the other hand, the main aim of visual information evaluation is to extract effective visual features and map them to comparable scores. Feature extraction is the key process, and is usually related to the application. For example, the visibility of a target [9], the location in an image or the real world [12, 22, 23], and the image area size [13] of the target are often used in tracking applications, whereas behavior analysis applications are more concerned with the orientation [33], body pose, and appearance [25, 27] of moving targets and the movement of the scene [7]. Surveillance applications for critical areas typically require front view, high-resolution video of the target of interest, and frequently switch views to prevent errors in visual computing. The front view of the target is determined by pose estimation and face detection; however, body pose estimation from a single perspective is still a nonlinear and ill-posed problem [1], and face detection has high error rates that are influenced by the video resolution and the relative position of the target with respect to the cameras. Therefore, the results of visual computing with both methods are often unstable and have high computational complexity.

In this paper, we address the problem of dynamic camera selection for the surveillance of critical areas. Our aim is to obtain the most informative camera node while simultaneously reducing camera switching. To achieve this, we model the dynamic selection of nodes in camera networks as a Markov Decision Process (MDP), and learn the selection strategy via reinforcement learning. For dynamic scenes, a state transition model is implicit, and the state space is continuous. We adopt an approximate Q-learning algorithm to find the optimal solution online, using a Gaussian Mixture Model (GMM) to represent the approximated Q-function. Model parameters are updated by a mini-batch stepwise Expectation–Maximization (EM) algorithm for the incoming episode state and Q-value. To evaluate the visual information, we design a function to integrate the visibility, orientation, and image resolution of the target, and conduct a trade-off between the video switching frequency and the amount of visual information.

The remainder of this paper is organized as follows. Section 2 discusses the problem of node selection in camera networks, before Section 3 presents our proposed selection policy-learning method based on Q-learning. Section 4 outlines how the Q-function is approximated with a GMM

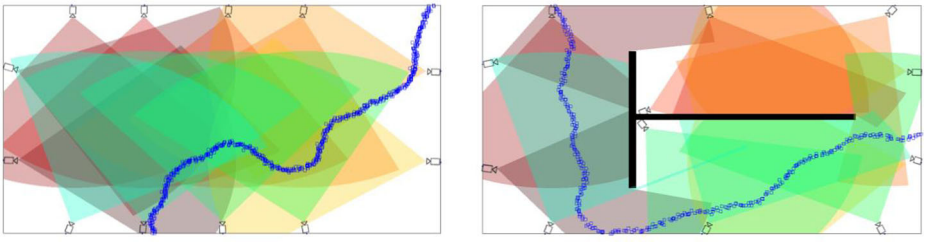
updated by mini-batch stepwise EM. Section 5 discusses visual information measures, and Section 6 presents the results of extensive experiments. Finally, we summarize this paper in Section 7.

2 Problem formulation

The camera node selection problem can be described as follows. There are N smart cameras $(C^1, C^2 \ldots C^N)$ consisting of an image sensor, a processor, and a wireless communication module with partially or completely overlapped FOVs, as illustrated in Fig. 1. We assume that the communication between the cameras is synchronized and delay-free. A designated central controller node schedules the cameras according to a selection policy. At a given time instant $t=0,1,2,\ldots$, the camera nodes in the network extract the target's visual features from their video streams, and convert them to a visual score that indicates the quality of the video stream. This score is then sent to the central controller. To obtain the maximum reward from the camera selection action, the central controller selects camera C^* as the optimal camera at that time instant according the previously selected camera and the visual scores. Therefore, camera C^* outputs its surveillance video for the interval $(t, t+\Delta t)$.

In the selection process, the central agent dynamically selects the video camera that can best obtain a high-quality surveillance video. Thus, it can be considered a sequential decision-making process based on the analysis and evaluation of visual content. Because the state transition and the expected reward only depend on the current system state and the previous selection result, this can be modeled as an MDP. An MDP can be formally defined as a 4-tuple $<A,S,T,R>$, where A is a finite set of candidate cameras $\{C^1, C^2 \ldots C^N\}$, S is the set of all possible underlying states, and the system state $\mathbf{s}_t \in S$ at time t is represented as a vector $\mathbf{s}_t=(P_t, O_t, a_{t-1})$, where P_t is the Δt target position $\{(x_t^1, y_t^1), (x_t^2, y_t^2), \ldots (x_t^{\Delta t}, y_t^{\Delta t})\}$ between the previous time point $t-1$ and the current point t, O_t is the sequence of target orientations $\{\theta_t^1, \theta_t^2, \ldots \theta_t^{\Delta t}\}$, and $a_{t-1} \in A$ is the node selected at time point t. T is a state transition function $T: S \times A \rightarrow S$, and $R(\mathbf{s}_t, a_t)$ is an immediate reward function $R: S \times A \rightarrow \mathbb{R}$ that assigns real-valued rewards to the selection actions performed in each of the underlying process states. To simplify the notation, we represent the immediate reward $R(\mathbf{s}_t, a_t)$ as r_t. Consequently, the MDP model can be used to derive a selection policy $\pi^*(\mathbf{s}): \mathbf{s} \mapsto a$ that yields the maximized expected reward $V^{\pi^*}(\mathbf{s})$ over some decision steps according to the history of states and actions, i.e.,

$$V^{\pi^*}(\mathbf{s}) = \max_{\pi^*} E\left[\sum_{t=0}^{\infty} \gamma^t r_t | \mathbf{s}_0 = \mathbf{s}\right] \tag{1}$$



(a) Scene A and trajectory of the target **(b)** Scene B and trajectory of the target

Fig. 1 Example of scenes

where $\gamma \in [0,1]$ is a discount factor that controls the impact of future rewards, causing the effect of a reward to decay exponentially with elapsed time. When the transition and reward functions are known, the model can be computed iteratively offline using dynamic programming. However, the transition model for the movement of the target is difficult to obtain for any real scene.

Reinforcement learning methods, in which agents interact with an unknown environment, synthesize, and improve their behavior through trial and error, have been extensively researched in recent years. We use a Q-learning-based algorithm to learn the optimal policy online, requiring no knowledge of the transition probabilities of the underlying MDP. The classic Q-learning algorithm maps every state-action pair to a real number (or Q-value) using a look-up table. The system described herein has a continuous state and discrete actions. Hence, a large storage space is needed to store all state-action pairs, and convergence is typically slow because it is not feasible to train all pairs within a large domain. Therefore, in this paper, we use a function approximated with a GMM and updated via stepwise EM for every episode sample to represent the Q-function.

3 Selecting camera-based Q-learning

3.1 Architecture of the proposed method

The general architecture of the node selection method for camera networks based on approximate Q-learning is depicted in Fig. 2. The key symbols used in this paper are given in Table 1. Target tracking is carried out by each camera node, and communication between cameras is used to enhance the fusion of the target state and association for a specific target. Visual scores are calculated and transferred to the central controller node, and the selected node then receives instructions from the central node to output its video stream.

The central controller has action and learning modules that can be executed in different parallel threads, and the Q-function $Q(\mathbf{s},a)$ is approximated by a group of

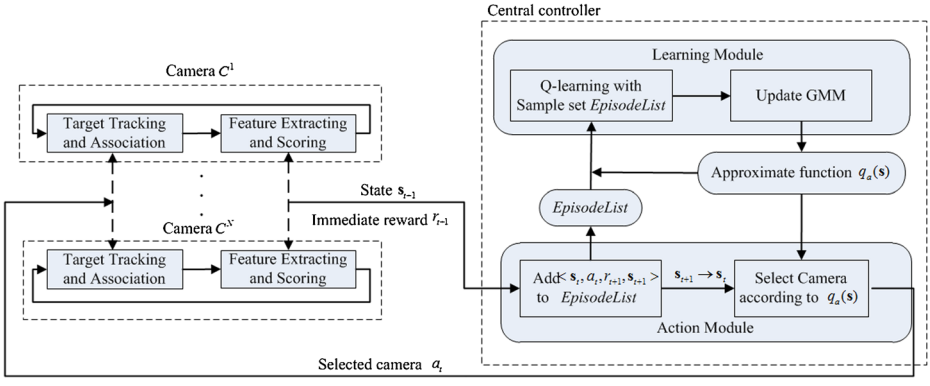


Fig. 2 Architecture of the proposed approximate Q-learning-based method

Table 1 Key symbols and notation

Symbol	Notation	Symbol	Notation
C^i	Camera i th	$\xi_{q_a,j}$	Sample j of Q-value
$\mathbf{s}_t$, a_t	State vector, selected camera at time t	K	Number of Gaussian components
r_t	Immediate reward at time t	D	Dimension of sample
π^*	Optimal selection policy	$\boldsymbol{\mu}^k$, $\boldsymbol{\Sigma}^k$	Mean, covariance of k th Gaussian component
$V^{\pi^*}(\mathbf{s})$	Expected long-term reward with policy π^* for state $\mathbf{s}$	η^k	Mixture weights of k th Gaussian component
γ	Discount factor for expected long-term reward	$\mathscr{L}_u$	Log-likelihood function for iteration u
$Q(\mathbf{s}_t,a_t)$	Q-value for $(\mathbf{s}_t,a_t)$	Θ	Parameter collection of Gaussian component
ρ	Learning rate for Q-Learning	$Suff_j^k$	Sufficient statistics triple for component k and sample data ξ_j
$q_{a_t}(\mathbf{s}_t)$	Approximate Q-function for $(\mathbf{s}_t,a_t)$	r_j^k	Update rate of sufficient statistics for component k and sample data ξ_j
$\hat{q}_{a_t}(\mathbf{s}_t)$	Sampled Q-value for state-action pair $(\mathbf{s}_t,a_t)$	i_k	Number of update samples for component k
Φ	Sample set for state-action pair and Q-value	β	Constant to scale variance of new component
(x_a,y_a)	Position of camera a	$\overline{q}(\mathbf{s})$	Conditional mean expectation of GMM for state $\mathbf{s}$
(x_s,y_s)	Position of the target	$\overline{\boldsymbol{\Sigma}}_{qq}$	Conditional covariance expectation matrix of GMM for state $\mathbf{s}$
θ_a	Angle between x -axis and the orientation of camera a	$\overline{M_{t+1}^{a_t}}$	Mean visual score
θ_s	Angle between x -axis and target's front side	M_v, M_d, M_c	Target visibility, orientation and clarity score
p_{a_t}	Probability of camera a_t being selected in exploration	w_0	Q-function initialization weight
ε_t	Exploration probability at time t	w_v, w_d, w_c	Weight of visual score, orientation score and clarity score
ε_0	Initial exploration probability	w_s	Weight of camera switching cost
ε_Δ	Exploration probability coefficient	ξ_j	Sample j of state and Q-value
τ	Weight coefficient for t	$\xi_{s,j}$	Sample j of state

GMM models in which $q_a(s), a=1\ldots N$ represents the expected Q-value for state $\mathbf{s}$ and camera a, that is, $Q(\mathbf{s},a)=E[q_a|\mathbf{s}]$. In the action module, the central agent selects camera node a_t at time step t using the action selection strategy presented in Section 3.4. To accelerate the learning, the target's current state and scene topology are used as a priori knowledge to initialize the Q-function, as described as Section 3.3. As a result, the selected camera index is transmitted to all nodes in the camera network, and the selected camera obtains the immediate reward r_{t+1} and observes the next state s_{t+1}. The new sample 4-tuple $<\mathbf{s}_t,a_t,r_{t+1},\mathbf{s}_{t+1}>$ is added to the sequence (or episode) $\mathbf{s}_0,a_0,r_1,\mathbf{s}_1,\ldots,\mathbf{s}_t,a_t,r_{t+1},\mathbf{s}_{t+1},\ldots,\mathbf{s}_{T-1},a_{T-1},r_T,\mathbf{s}_T$. Each episode ends when the target leaves the monitoring area or the number of selection actions reaches some

pre-assigned value. In our method, a collection referred to as the *EpisodeList* is used to store the episodes produced by the action module. The *EpisodeList* is sent to the learning module to provide training samples when the number of episodes reaches a certain threshold.

In the learning module, the Q-value samples can be trained with the 4-tuple samples in *EpisodeList* and the current $q_a(\mathbf{s})$ by the Q-learning algorithm, described in Section 3.2. The parameters of the GMM for the cameras can then be updated online via mini-batch stepwise EM with the Q-value samples. The new approximation function $q'_a(\mathbf{s})$ is then input to the action module as the basis of camera selection.

3.2 Q-learning

In reinforcement learning, an agent generally receives a reward for actions that are beneficial along the path to achieving some long-term goal. Q-learning is a well-known model-free reinforcement learning method that estimates the optimal action-value function [34] with no explicit knowledge of the dynamic environment. In this paper, every state-action pair is mapped to a real number by the function $Q(\mathbf{s}_t, a_t)$, which is an estimated incremental function for the state-action pair $(\mathbf{s}_t, a_t)$ based on the temporal difference principle in Q-learning, i.e.,

$$Q(\mathbf{s}_t, a_t) \leftarrow Q(\mathbf{s}_t, a_t) + \rho\left[r_{t+1} + \gamma \max_a Q(\mathbf{s}_{t+1}, a) - Q(\mathbf{s}_t, a_t)\right] \tag{2}$$

where $\rho \in [0,1]$ is the learning rate controlling how fast the Q-function estimations are modified (we set ρ=0.1), γ is the discount factor for expected long-term rewards, and camera node a is selected to maximize the discounted sum of rewards $\max_a Q(\mathbf{s}_{t+1}, a)$. Q-learning converges to the optimal Q-function if each state-action pair is performed infinitely often and ρ is satisfied for each $(\mathbf{s}_t, a_t)$ pair: $\sum\rho = \infty$ and $\sum\rho^2 < \infty$ [34]. When the Q-function converges, the optimal camera selection policy π(s) can be obtained by maximizing the function, i.e.,

$$\pi(\mathbf{s}) = \arg\max_{a \in A} Q(\mathbf{s}, a) \tag{3}$$

We assume there is some approximate function $q_a(\mathbf{s})$ that corresponds to each camera a such that the Q-value for the state-action pair $(\mathbf{s}, a)$ is the expectation of the function $q_a(\mathbf{s})$ for state $\mathbf{s}$, i.e.,

$$Q(\mathbf{s}, a) = E[q_a(\mathbf{s})] = \mu[q_a(\mathbf{s})] \tag{4}$$

For every piece of sample data in *EpisodeList*, an initial Q-value sample $< \mathbf{s}_t, a_t, \hat{q}_{a_t}(\mathbf{s}_t) >$ is calculated for the state-action pair $(\mathbf{s}_t, a_t)$, and the current approximate function is given by Gaussian Mixture Regression with the corresponding GMM model. The Q-value $\hat{q}_{a_t}(\mathbf{s}_t)$ is iteratively updated using:

$$\hat{q}_{a_t}(\mathbf{s}_t) \leftarrow \hat{q}_{a_t}(\mathbf{s}_t) + \rho\left[r_{t+1} + \gamma \max_a q_a(\mathbf{s}_{t+1}) - \hat{q}_{a_t}(\mathbf{s}_t)\right] \tag{5}$$

When the iterative learning has been completed for all samples in *EpisodeList*, the state $\mathbf{s}_t$, selected camera a_t, and Q-value $\hat{q}_a(s)$ are added to the Q-value sample collection $\Phi =$

$\{< s_t, a_t, \hat{q}_{a_t}(s_t) >\}$. The subset $D_a = \{< s_t, a_t, \hat{q}_{a_t}(s_t) > | a = a_t\}$ for camera a_t is used to update the GMM approximation function for the corresponding camera. The pseudo-code for this process is outlined in Algorithm 1.

Algorithm 1: Q-learning

Q-learning Procedure

Input: *EpisodeList*, GMM , γ , ρ

initialize Q-function sample collection $\Phi = \varnothing$
for $l = 1...L$ times (the time of the iteration)
  for $t = 0,...,T-1$ (the time steps of the episode)
    retrieve the sample $< s_t, a_t, r_{t+1}, s_{t+1} >$ from *EpisodeList*
    if $l = 1$ **then**
      compute $\hat{q}_{a_t}(s_t)$ with GMM
      $\Phi \leftarrow \Phi \bigcup \{< s_t, a_t, \hat{q}_{a_t}(s_t) >\}$
    end if
    $\hat{q}_{a_t}(s_t) \leftarrow \hat{q}_{a_t}(s_t) + \rho[r_{t+1} + \gamma \max_a q_a(s_{t+1}) - \hat{q}_{a_t}(s_t)]$
    Update $< s_t, a_t, \hat{q}_{a_t}(s_t) >$ in Φ
  end for
end for
end procedure
output Φ

As can be seen from Algorithm 1, the immediate reward r_{t+1} is the visual score of the target between t and $t+\Delta t$, as outlined in Section 5.

3.3 Q-function initialization

Q-learning starts from an arbitrary initial Q-function $Q_0(\mathbf{s}, a)$ that can be updated without requiring an a priori model. Thus, Q-learning often suffers from slow convergence when there are a large number of states and action spaces. Considering that the camera node with the maximum visual score can be selected greedily if the target's state remains stable in a subsequent period of time, we use the target's current state and scene topology as a priori knowledge, and initialize $Q_0(\mathbf{s}, a)$ with the immediate reward for greedy selection in state s_0. This accelerates the convergence speed of the learning. Assuming that the hardware configuration of the cameras in the scene is the same, the immediate reward for greedy selection is associated with the relative position and angle between the target and the selected camera. Thus, the initial pair value $Q_0(\mathbf{s}, a)$ can be defined as

$$Q_0(s,a) = w_0\left(\sin\left(\frac{|\theta_s - \theta_a|}{2}\right)\right) + (1-w_0)\left(1 - \frac{\sqrt{(x_s - x_a)^2 + (y_s - y_a)^2}}{MaxDis}\right) \tag{6}$$

where w_0 is a predefined weight, (x_s,y_s) and (x_a,y_a) give the position of the target and camera a, *MaxDis* is the maximum distance to normalization, and θ_a, θ_s are the angles of offset from the x -axis in camera a and the target's front side, respectively, in state **s**. This equation implies that a closer distance and smaller angle between the camera and the target will result in better-quality monitoring videos.

3.4 Action selection strategy

In reinforcement learning, action selection must address the trade-off between exploration (that is, exploring the world to gather information about the system) and exploitation (which involves executing actions with the current policy). To obtain a greater initial exploration probability that gradually decreases with time until a steady state is reached, we implement an improved ε-*greedy* action selection policy [32], where greedy action a_t is selected with a probability of $1-\varepsilon_t$, i.e., $a_t=\arg\max_a Q(s_t,a)$, whereas a non-greedy action is selected with probability ε_t. We set ε_t as the sum of the initial exploration probability ε_0 and an exponential function of time t produced with exploration probability coefficient ε_Δ, i.e.,

$$\varepsilon_t = \varepsilon_0 + \varepsilon_\Delta e^{-t/\tau} \tag{7}$$

where the weight coefficient τ is set to 300 in this study. In the case of a non-greedy action, the probability of selecting a candidate camera with a larger Q-value is enhanced by using a Gibbs distribution to select the exploratory action instead of a random distribution. That is, at time t during the exploration, camera a_t is selected with probability

$$p_{a_t} = \frac{e^{q_a(s_t)\varepsilon_t}}{\sum_{b=1}^{N} e^{q_a(s_t)\varepsilon_t}} \tag{8}$$

where N is the number of cameras in the scene.

4 Q-function approximation

Compared to the full-state representation using Q-values, the function approximation method can reduce the required storage space and accelerate convergence by exploiting the similarity of Q-values between adjacent state-action pairs in the continuous state. Function approximation methods in reinforcement learning include linear parameter representations [28], fuzzy rules [4], neural networks [3], and kernel-based techniques [14]. Approximation based on GMMs is a non-parametric function approximation method [2] that can represent complex systems by changing the number of Gaussian components. In this paper, we present an approximate Q-function method based on GMMs, whereby the GMM parameters are sequentially updated by the mini-

batch stepwise EM algorithm. The procedure is divided into offline model initialization and online updating. In the initialization stage, we sample randomly from the state space, and first set the Q-value as the score of the corresponding state, as described in Section 3.3. The initial model parameters are then obtained by offline stepwise EM. In the online update stage, the set of samples Φ learnt from Q-learning are divided into blocks of size m, and the online stepwise EM iterates over each block to update the approximate model parameters.

4.1 GMMs

GMMs are a popular method for representing general probability density functions in multidimensional spaces by the weighted sum of some Gaussian component densities [20], with the number of components adjusted according to the complexity of the problem. In this paper, we assign each camera a GMM, and define the training set of the approximate Q-function for camera a as $\Phi_a = \{\boldsymbol{\xi}_j\}_{j=1}^{J} = \left\{\boldsymbol{\xi}_{s,j}, \xi_{q_a,j}\right\}_{j=1}^{J}$ (written as Φ for simplicity), where $\boldsymbol{\xi}_{s,j}=\mathbf{s}_j$, $\xi_{q_a,j} = \hat{q}_a^j$, and J is the number of samples. The probability distribution of $\boldsymbol{\xi}_j$ is then the weighted sum of K Gaussian densities, given by:

$$p(\xi_j) = \sum_{k=1}^{K} \eta^k N(\xi_j; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k) \tag{9}$$

$$N(\boldsymbol{\xi}_j; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k) = \frac{1}{(2\pi)^{D/2}\left|\boldsymbol{\Sigma}^k\right|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{\xi}_j-\boldsymbol{\mu}^k)'\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{\xi}_j-\boldsymbol{\mu}^k)\right\} \tag{10}$$

with mean vector $\boldsymbol{\mu}^k$ and covariance $\boldsymbol{\Sigma}^k$ for Gaussian component k; where D is the dimension of the samples, and $\eta^k, k=1\ldots K$ are the mixture weights that satisfy the constraint $\sum_{k=1}^{K} \eta^k = 1$. Therefore, these parameters are collectively represented by the notation $\Theta=\{\eta^k, \boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k\}_{k=1}^{K}$. For the training set Φ, the log-likelihood function is defined as:

$$\mathscr{L}(\Phi;\Theta) = \ln \prod_{j=1}^{J} p(\boldsymbol{\xi}_j;\Theta) = \sum_{j=1}^{J} \ln\left(p(\boldsymbol{\xi}_j;\Theta)\right) \tag{11}$$

The GMM parameters are traditionally trained in batch mode (i.e., using the whole training set) using the iterative EM algorithm. An initial model Θ_0 is generated by sampling randomly in the state space to give a rough estimation, and the *uth* iterative model $\Theta_u=\{\eta_u^k, \boldsymbol{\mu}_u^k, \boldsymbol{\Sigma}_u^k\}_{k=1}^{K}$ is estimated via E-steps and M-steps until convergence:

E-step:

$$p_{u+1}\left(k\middle|\boldsymbol{\xi}_j\right)=\frac{\eta_u^k N\left(\boldsymbol{\xi}_j;\boldsymbol{\mu}_u^k,\boldsymbol{\Sigma}_u^k\right)}{\sum_{k=1}^{K}\eta_u^k N\left(\boldsymbol{\xi}_j;\boldsymbol{\mu}_u^k,\boldsymbol{\Sigma}_u^k\right)} \tag{12}$$

$$E_{u+1}^k=p_{u+1}(k)=\sum_{j=1}^{J}p_{u+1}\left(k\middle|\boldsymbol{\xi}_j\right) \tag{13}$$

M-step:

$$\eta_{u+1}^k=\frac{1}{J}E_{u+1}^k \tag{14}$$

$$\mu_{u+1}^k=\frac{\sum_{j=1}^{J}p_{u+1}\left(k\middle|\boldsymbol{\xi}_j\right)\boldsymbol{\xi}_j}{E_{u+1}^k} \tag{15}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{u+1}^k&=\frac{\sum_{j=1}^{J}p_{u+1}\left(k\middle|\boldsymbol{\xi}_j\right)\left(\boldsymbol{\xi}_j-\boldsymbol{\mu}_{u+1}^k\right)\left(\boldsymbol{\xi}_j-\boldsymbol{\mu}_{u+1}^k\right)^T}{E_{u+1}^k}\\&=\frac{\sum_{j=1}^{J}p_{u+1}\left(k\middle|\boldsymbol{\xi}_j\right)\boldsymbol{\xi}_j\boldsymbol{\xi}_j^T}{E_{u+1}^k}-\boldsymbol{\mu}_{u+1}^k\left(\boldsymbol{\mu}_{u+1}^k\right)^T\end{aligned} \tag{16}$$

The iteration terminates when $\left|\frac{\mathscr{L}_{u+1}}{\mathscr{L}_u}-1\right|<C_1$, where C_1 is a small number. It is worth noting that the GMM parameters η_{u+1}^k, $\boldsymbol{\mu}_{u+1}^k$, and $\boldsymbol{\Sigma}_{u+1}^k$ can be regarded as the weighted means of the sufficient statistics for the triple $\{1,\boldsymbol{\xi}_j,\boldsymbol{\xi}_j\boldsymbol{\xi}_j^T\}$ with the posterior probability $p_{u+1}(k|\boldsymbol{\xi}_j)$.

4.2 Stepwise EM

Because the approximate function trainer obtains the training episode data in multiple batches, the GMM needs to be sequentially updated online to conserve storage space and accelerate convergence. In this paper, we utilize the stepwise EM algorithm [19, 26] to sequentially update the GMM parameters in mini-batch mode with the Q-value of the sampling episode. In each update, the new Q-value samples, initial model parameters, and the number of historic

update samples for each component are provided as input, and the algorithm is divided into E-steps and M-steps. Finally, the updated model parameters and the number of historic update samples are output. The pseudo-code for online stepwise EM is outlined in Algorithm 2.

Algorithm2: Online stepwise EM

Online stepwise EM procedure
Input: $\Phi, \Theta_{init}, \{i_1,...,i_K\}$

$\Theta_{old} \leftarrow \Theta_{init}$
for $j = 1,\ldots,J$
  //E-step:
  for $k = 1,\ldots,K$
    $p(k \mid \boldsymbol{\xi}_j) \leftarrow \text{Posterior}(\boldsymbol{\xi}_j, \Theta_{old}^k)$ //eq.(12)
  end for
  if $\forall N(\boldsymbol{\xi}_j; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k) < ThredNew$ **then**
    create a new component;
    $K = K+1; i_K \leftarrow 1;$
  else
    for $k = 1,\ldots,K$
      $Suff_j^k \leftarrow UpdateSufficientStatistics(\boldsymbol{\xi}_j, Suff_{j-1}^k)$ //eq.(17)
      $i_k \leftarrow i_k + 1$
    end for
  end if
  if $j \bmod m = 0$ **then**
    //M-step:
    $\Theta_{new} \leftarrow \overline{\Theta}(Suff_j^1,\ldots Suff_j^K)$ //eq.(18)-eq.(20)
    $\Theta_{old} \leftarrow \Theta_{new}$
  end if
end for
end procedure
output $\Theta_{new}, \{i_1,...,i_K\}$

In the E-step of Algorithm 2, the posterior probability $p(k|\boldsymbol{\xi}_j)$ of Gaussian component k for sample $\boldsymbol{\xi}_j$ is calculated using (12). If all posterior probabilities of the GMM components are less than the threshold *ThredNew*, the new sample is deemed to be too far from the present GMM component to be explained by the current stochastic model. In this case, a new component is produced to account for the new sample data and added to the model. Otherwise, a sufficient statistics triple $Suff_j^k = \{<<1>>_j^k, <<\boldsymbol{\xi}>>_j^k, <<\boldsymbol{\xi}\boldsymbol{\xi}^T>>_j^k\}$ for the *kth* component is stepwise updated with sample data $\boldsymbol{\xi}_j$, in which the component $<<f(x)>>_j^k$ is a time-discounted weighted sum defined as

$$<< f(x) >>_j^k = \left(1 - r_j^k\right) << f(x) >>_{j-1}^k + r_j^k p\left(k \middle| \boldsymbol{\xi}_j\right) f(x) \tag{17}$$

where $\{r_j^k\}_{j\geq 1}$ is a decreasing sequence with the number of historic update samples i_k for component k, which should satisfy $\sum_j r_j^k=\infty$ and $\sum_j (r_j^k)^2<\infty$ (we set $r_j^k=(i_k+2)^{-\alpha}$), and $\alpha\in[0.6, 0.9]$.

In the M-step, to stabilize the algorithm, we update m samples at once; more specifically, the training set is divided into mini-batches. Thus, for sample $\boldsymbol{\xi}_j$ ($j \bmod m=0$), the update function $\overline{\Theta}\left(Suff_j^1, \ldots Suff_j^K\right)$ for the model parameters is defined as:

$$\overline{\Theta}: \qquad \eta_j^k = \frac{<<1>>_j^k}{\sum_{k=1}^{K} <<1>>_j^k} \tag{18}$$

$$\boldsymbol{\mu}_j^k = \frac{<<\boldsymbol{\xi}>>_j^k}{<<1>>_j^k} \tag{19}$$

$$\boldsymbol{\Sigma}_j^k = \frac{<<\boldsymbol{\xi}\boldsymbol{\xi}^T>>_j^k}{<<1>>_j^k} - \boldsymbol{\mu}_j^k \boldsymbol{\mu}_j^{k\prime} \tag{20}$$

Initialization of GMMs Each model component is initialized prior to sequential updating. First, we conduct random sampling in the state space, and set the initial Q-value for the states, as described in Section 3.3. Algorithm 2 is then executed iteratively to update the components of the GMM according to the camera ID in the state sample until $\left|\frac{\mathscr{L}_{u+1}}{\mathscr{L}_u}-1\right| < C_1$ or the maximum number of iterations for (11) is reached.

Component production We use a minimum likelihood criterion to recognize a new sample as belonging to a mixture component. For the incoming sample, the algorithm verifies whether it minimally fits any component, and then decides whether a new Gaussian component should be produced and added to the model. Because the probability $N(\boldsymbol{\xi}_j; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k)$ is interpreted as a likelihood function of the *kth* component for sample $\boldsymbol{\xi}_j$, sample $\boldsymbol{\xi}_j$ is not recognized as belonging to any component in the model if its probability $N(\boldsymbol{\xi}_j; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k)$ is lower than the minimum likelihood threshold of all components, i.e.,

$$N\left(\boldsymbol{\xi}_j; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k\right) < \frac{\tau_{\min}}{(2\pi)^{D/2}\sqrt{|\boldsymbol{\Sigma}_k|}} \forall k \tag{21}$$

where $\tau_{\min}$ is a previously specified acceptable likelihood for a fraction of the minimum likelihood value, making the minimum likelihood criterion independent of the covariance matrix. If the sample is rejected by all Gaussian components of the model for the corresponding camera, then it is considered to include a new concept. In this case, a new component $K+1$ is added to the model, and the initial parameters of the new component are given by

$$\eta_j^{K+1} = \frac{1}{1+\sum_{k=1}^{K} <<1>>_j^k} \tag{22}$$

$$\boldsymbol{\mu}_j^{K+1} = \boldsymbol{\xi}_j \tag{23}$$

$$\boldsymbol{\Sigma}_j^{K+1} = \beta diag(d_1, d_2, \ldots, d_D) \tag{24}$$

where $d_1, d_2, \ldots, d_D$ in (24) are the ranges of variances, and β is an appropriate positive constant for scaling these variances. To control the number of Gaussian components and ensure that the complexity is within a certain range, we set the maximum number of Gaussian components to MAX_{comp}. When a new component is required and $K+1 > MAX_{comp}$, the most infrequently updated component with the minimum sufficient statistics $<<1>>_j^k$; i.e., $\arg\min_{k=1}^{K} <<1>>_j^k$, is replaced by the new component.

4.3 Gaussian mixture regression

The approximate Q-value is calculated with the current state and GMM in the action module. To obtain the initial Q-value $\hat{q}_{a_t}(s_t)$ and $\max_a q_a(s_{t+1})$ in (5), we estimate the expectation of the Q-value for state $\mathbf{s}$ using Gaussian Mixture Regression (GMR) [6]. In a generic regression problem, we are given a set of predictor variables $X \in \mathbb{R}^p$ and response variables $Y \in \mathbb{R}^g$, and the aim of GMR is to estimate the conditional expectation of Y given X on the basis of a set of observations $\{X, Y\}$. For the GMM model Θ that encodes the sample set $\boldsymbol{\xi}_j = \left\{\boldsymbol{\xi}_{s,j}, \xi_{q_a,j}\right\}$, the state and Q-value of the Gaussian component k for camera a are also separated, i.e., we define

$$\boldsymbol{\mu}_k = \left\{\boldsymbol{\mu}_{s,k}, \mu_{q,k}\right\}, \boldsymbol{\Sigma}_k = \begin{pmatrix} \boldsymbol{\Sigma}_{ss,k} & \boldsymbol{\Sigma}_{sq,k} \\ \boldsymbol{\Sigma}_{qs,k} & \boldsymbol{\Sigma}_{qq,k} \end{pmatrix} \tag{25}$$

Therefore, the conditional probability of the Q-value q is the mixture of the probability that component k is responsible for state $\mathbf{s}$; specifically,

$$p\left(q\middle|\mathbf{s}\right) = \sum_{k=1}^{K} \beta_k p\left(q_k\middle|\mathbf{s}, k\right) = \sum_{k=1}^{K} \beta_k N\left(q_k; \overline{q}_k(\mathbf{s}), \overline{\boldsymbol{\Sigma}}_{qq,k}\right) \tag{26}$$

Where

$$\overline{q}_k(\mathbf{s}) = \mu_{q,k} + \boldsymbol{\Sigma}_{qs,k}\left(\boldsymbol{\Sigma}_{ss,k}\right)^{-1}\left(\mathbf{s} - \boldsymbol{\mu}_{s,k}\right) \tag{27}$$

$$\overline{\boldsymbol{\Sigma}}_{qq,k} = \boldsymbol{\Sigma}_{qq,k} - \boldsymbol{\Sigma}_{qs,k}\left(\boldsymbol{\Sigma}_{ss,k}\right)^{-1}\boldsymbol{\Sigma}_{sq,k} \tag{28}$$

$$\beta_k = \frac{p(k)p\left(s\middle|k\right)}{\sum_{i=1}^{K} p(i)p\left(s\middle|i\right)} = \frac{\eta_k N\left(s; \mu_{s,k}, \boldsymbol{\Sigma}_{ss,k}\right)}{\sum_{i=1}^{K} \eta_i N\left(s; \mu_{s,i}, \boldsymbol{\Sigma}_{ss,i}\right)} \tag{29}$$

Finally, as a mixture of K Gaussian components, the estimation of the conditional expectation of $q(s)$ given s (that is, the conditional mean expectation $\overline{q}(\mathbf{s})$) and conditional covariance expectation matrix $\overline{\Sigma}_{qq}$ are computed using (26) and

$$\overline{q}(\mathbf{s}) = \sum_{k=1}^{K} \beta_k \overline{q}_k(\mathbf{s}) \tag{30}$$

$$\overline{\Sigma}_{qq} = \sum_{k=1}^{K} \beta_k^2 \overline{\Sigma}_{qq,k} \tag{31}$$

Thus, by evaluating $\left\{\overline{q}(\mathbf{s}), \overline{\Sigma}_{qq}\right\}$ for different states $\mathbf{s}$, we can obtain the generalized form of data point $\hat{\xi} = \{\mathbf{s}, \overline{q}(\mathbf{s})\}$ with the associated covariance matrix $\overline{\Sigma}_{qq}$, which gives the estimated Q-value.

5 Measures of visual information

In this paper, the immediate reward described in Section 3 is presented as a set of metrics that evaluate the quality of visual information. To reflect the target information in the video and the visual comfort level of the output video stream, the reward r_{t+1} is composed of $\overline{M_{t+1}^{a_t}}$, the average visual score per frame for the target from two adjacent selections (positive reward), and the cost for view switching (negative reward). This can be expressed as follows:

$$r_{t+1} = w_v \overline{M_{t+1}^{a_t}} - w_s \delta(a_t, a_{t+1}) \tag{32}$$

$$\overline{M_{t+1}^{a_t}} = \frac{1}{T} \sum_{k=t+1}^{t+T} M_k^{a_t} \tag{33}$$

$$\delta(a_t, a_{t+1}) = \begin{cases} 0 & a_t = a_{t+1} \\ 1 & otherwise \end{cases} \tag{34}$$

In (32), $w_v, w_s \in [0,1]$ is a normalized weight for the visual score and switching, $\overline{M_{t+1}^{a_t}}$ is the mean visual score of the selected camera a_t for T frames after the selection time t in (33), and the switching cost is expressed as a δ function $\delta(a_t, a_{t+1})$ in (34). Further, for the selected camera a_t, the visual score of the target at time t is composed of the visibility score $M_v^{a_i}$, normalized orientation score $M_d^{a_i}$, and the clarity score $M_c^{a_i}$; that is,

$$M^{a_i} = M_v^{a_i}\left(w_d M_d^{a_i} + w_c M_c^{a_i}\right) \tag{35}$$

In (35), w_d and w_c ($w_d + w_c = 1$) are the weights of the orientation and clarity scores, respectively. They define the importance of each metric, and can be learned and modified. Higher the weight, greater the emphasis on the corresponding metric. To simplify the problem, we represent the target visibility score, orientation, and image resolution as being camera-independent; that is, M_v, M_d, and M_c.

In the process of calculating the visual information score, we first track the target in the selected camera's video, and detect the target's bounding box. We then obtain the target's trajectory P_{t+1} between time t and $t+1$ in the ground plane, using the bottom center of the bounding boxes and the calibration parameters of the camera. The metrics for measuring the visual information are outlined below.

Visibility The ability of a camera to capture the target is a crucial metric for camera selection. In this paper, we infer the visibility of the target for a particular camera according to the maximum area of the specified target's motion region in the video frames from human motion detection. That is, when the area of motion is smaller than a predetermined threshold, it can be inferred that the target is invisible, i.e., $M_v=0$; otherwise, $M_v=1$.

Orientation We infer the orientation from the trajectory of the target. Suppose that the target's front is consistent with its direction of movement, and its trajectory can be considered as a straight line over a short period of time. The orientation angle $\theta_t^{\Delta t}$ of the target at time $t+\Delta t$ for the target position sequence $P_{t+1}=\{(x_t^1,y_t^1),(x_t^2,y_t^2),\ldots(x_t^{\Delta t},y_t^{\Delta t})\}$ can be approximated by the slope of the line fitted from the closest l points $\{(x_t^{\Delta t-l+1},y_t^{\Delta t-l+1}),(x_t^{\Delta t-l+2},y_t^{\Delta t-l+2}),\ldots(x_t^{\Delta t},y_t^{\Delta t})\}$, as illustrated in Fig. 3. Further, fitting such a line can reduce the errors introduced by tracking and calibration. As in (6), the orientation score of the selected camera node at time $t+\Delta t$ can be defined as

$$M_d = \sin\left(\frac{|\theta_t^k-\theta_a|}{2}\right) \tag{36}$$

The above equation implies that the more positive the direction of motion with respect to the selected camera, the larger the value of M_d. This score reaches a maximum when the angle

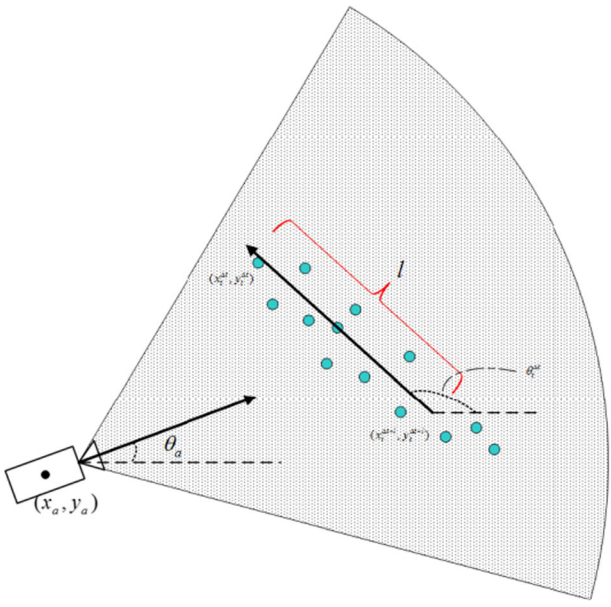


Fig. 3 Orientation angle based on a target's position sequence

between the direction of motion and the orientation of the camera is 180°, that is, the camera is facing forward towards the target's front.

Clarity The image clarity of a target reflects the ability of the camera to obtain detailed information under different perspectives, and can usually be represented by an average gradient or information entropy of the image region. Because the size of a target's image will vary from different viewpoints, and to compare the resolutions from different views for the same target, we zoom the target region image to the same height H. If the width of the scaled image is W, we represent the mean gradient of the zoomed image as the clarity score M_c for camera c, i.e.,

$$M_c = \frac{1}{(W-1)(H-1)} \sum_{x=1}^{W-1} \sum_{y=1}^{H-1} \sqrt{\left(\frac{\partial I(x,y)}{\partial x}\right)^2 + \left(\frac{\partial I(x,y)}{\partial y}\right)^2} \tag{37}$$

where $I(x,y)$ is the gray value at point (x,y).

6 Experiments and results

6.1 Experimental setup

We conducted simulations and actual scene experiments using an Intel Core i7-3770 2.4 GHz CPU PC with 8 GB RAM. Section 6.2 discusses the simulation experiments, including various scenes and camera parameters under noisy conditions. We used simulations to compare the correctness and convergence of the proposed algorithm with that of greedy selection and classic Q-learning algorithms because simulations can effectively compensate for the limits of hardware and space in real experiments. The convergence rate using the Q-value function initialized as in Section 3.3 was compared with that of the traditional method, which sets the initial Q-value to zero. We also analyzed various effects of the selection results with different time intervals by comparing greedy selection (Max), game theory [16] (Game), and POMDP [31] with the proposed method. These three comparative methods are based on maximizing the current visual information, optimization, and scheduling policies, and are convenient for comparing performance within the framework of scoring and selection. In Section 6.3, the visual features proposed in Section 5 are evaluated and the selection performance of Max, Game, POMDP, and our method are compared. In our method, the learning rate and exploration probability for Q-learning were determined by the values of ρ=0.1, ε_0= 0.1, and ε_Δ=0.2 according to the general reinforcement learning method. The discount factor in (2) was set to γ=0.85, which satisfies γ^T<0.02 and ensures that the currently selected action will not affect the state after T steps for T>10. For the Q-function approximation, our experiments show that better iteration results can be achieved when the stepwise weight exponent α is set to 0.8 and the maximum number of GMM components is set to MAX_{comp}=30, with the initial number set to 10. Moreover, we set the scaling constant to variance of new component β as 0.6 and the acceptable likelihood $\tau_{\min}$=0.02. As described in Section 5, the visual feature weights reflect the degree of importance of each feature in different applications; here, we set the weights to w_v=0.5, w_s=0.5, w_d=0.6, and w_c=0.4. To increase the stability of line-fitting when computing the orientation, we set the number of trajectory points involved in fitting l to 10, and the height of the scaled image region of the target H to 300.

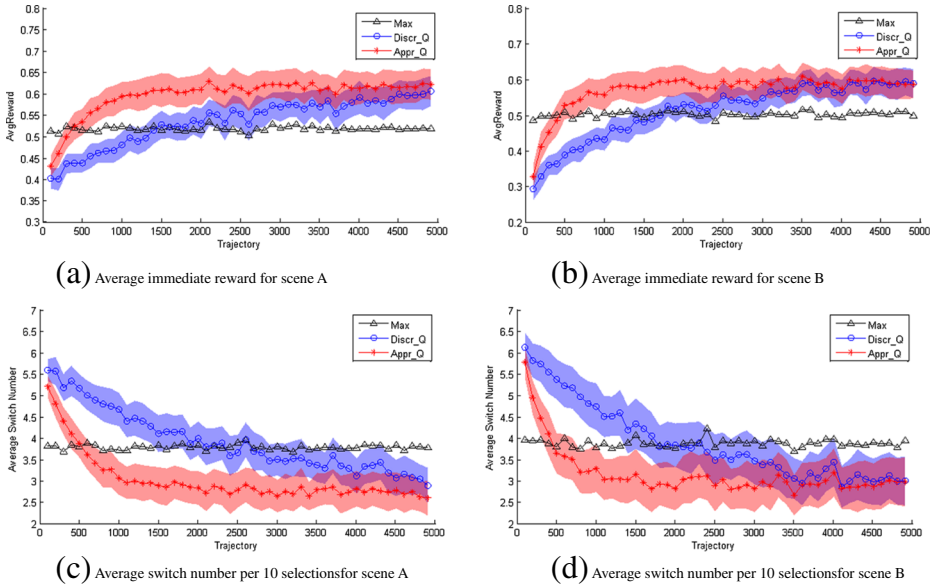
6.2 Results of simulation experiments

We conducted a number of simulation experiments by deploying 12 static cameras in a virtual rectangular region of length 12 m, width 8 m, and height 2 m. In this paper, we analyze the simulation results for example scenes A and B shown in Fig. 1. The cameras were deployed at the edge of scene A with overlapped or partially overlapped FOVs. In scene B, there were some static obstacles that necessitated handoffs between cameras. 20,000 trajectories were sampled in each scene to form a training dataset. In these trajectories, the target states consisted of the position coordinates (x,y) and orientation angles θ, which were obtained from the positions in the previous time step, i.e., $\theta=\arctan((y_t-y_{t-1})/(x_t-x_{t-1}))$. To simulate a more realistic data process, Gaussian noise with mean zero and covariance σ was added as observation noise. To evaluate the visual information, we adopted the visibility, orientation, and clarity metrics proposed in Section 5. The visibility in scene A is defined as the angle between the target-camera line and the center axis being less than a certain threshold and the distance between the target and camera line being less than the radius of the FOV. In scene B, we additionally consider whether the target-camera line intersects the region of occlusion. If the camera configuration in both scenes is the same, the resolution of the target image in the simulation experiments can be represented by the distance between the target and the camera; that is, the clarity score for a target at (x_s,y_s) and camera at $\left(x_{c_i},y_{c_i}\right)$ is $M_c^{a_i}=1-\sqrt{\left(x_s-x_{a_i}\right)^2+\left(y_s-y_{a_i}\right)^2}/MaxDis$, which is similar to (6).

To analyze the effectiveness and convergence, we used the 20,000 trajectories to train greedy selection (Max), discrete Q-learning (Discr_Q), and the proposed method (Appr_Q) 10 times for each scene. In the greedy selection method, the camera with the maximum visual score was selected. In the discrete Q-learning method, the system state space was discretized into 60×40×8 grids; more specifically, the virtual region was represented as a 60×40 grid according to the length and width, and the orientation angle was denoted as one of eight states, each covering 45°. Each method executed a selection decision at intervals of $\Delta t=20$ samples. The average immediate reward $Avg\mathrm{Re}ward=\frac{1}{T}\sum_{t=1}^{T} r_t$, where T is the total number of total selection points, was calculated for each trajectory, and the average number of camera switches $AvgSwitchNum=\frac{10}{T}\sum_{t=1}^{T}\delta\left(a_t,a_{t+1}\right)$ was computed for each group of 10 selections.

The mean *AvgReward* curves for scenes A and B are shown in Fig. 4a and b, where the shaded areas represent the standard deviation of discrete Q-learning and approximate Q-learning, respectively, for 10 training sessions. Further, the *AvgSwitchNum* curves for scenes A and B are shown in Fig. 4c and d, where shaded areas around the discrete Q-learning and approximate Q-learning curves represent the standard deviation over 10 training sessions. As can be seen from Fig. 4, reinforcement learning methods outperform greedy selection in terms of immediate reward and switching number after they have converged. Further, the approximate function method converges faster than discrete Q-learning. In addition, the approximate method is marginally better than the discrete method in terms of both reward and switching number, because it is able to represent the model states more compactly.

We initialized the Q-value with the GMM in accordance with Section 3.3 (*Initiation_Appr_Q*) and compared the results to those when the initial Q-value was set to zero (*Without_Initiation_Appr_Q*) and discrete Q-Learning (*Without_Initiation_Discr_Q*), for 60,000 trajectories and 10 training sessions. The mean curves for *AvgSwitchNum* and



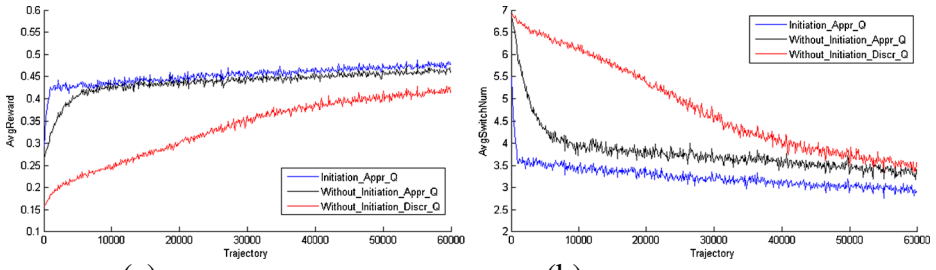
(a) Average immediate reward for scene A (b) Average immediate reward for scene B

(c) Average switch number per 10 selectionsfor scene A (d) Average switch number per 10 selectionsfor scene B

Fig. 4 Average reward and switch number for scenes A and B (see Fig. 1)

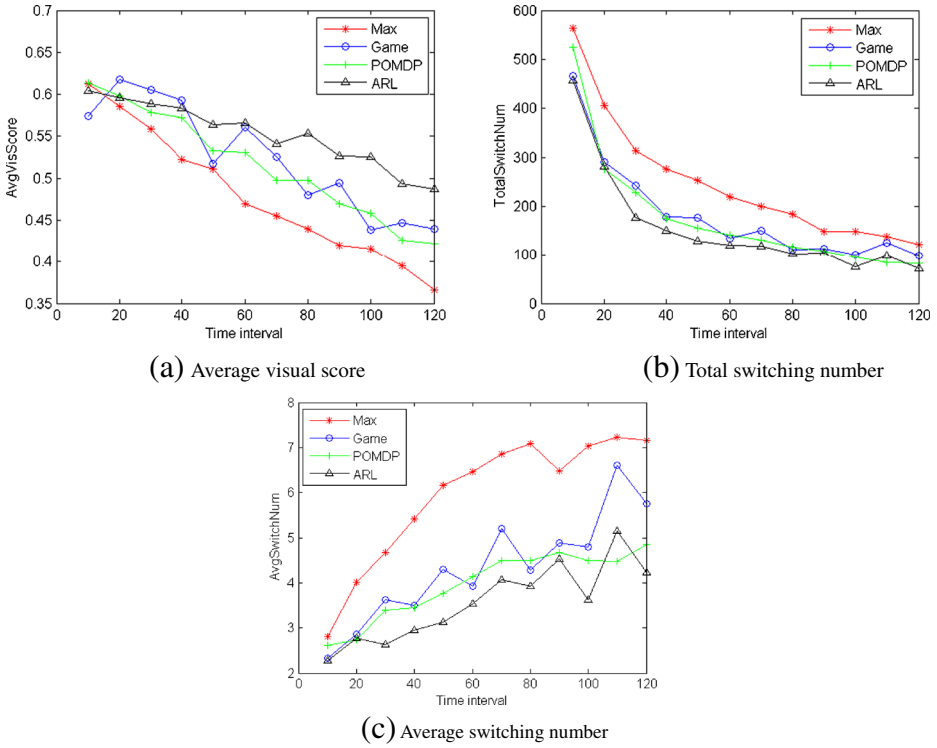
AvgReward are shown in Fig. 5. The figure shows that the Q-value initialized with the target location and camera parameters tends to produce convergence after approximately 500 trajectories. In contrast, the Q-values initialized to zero for approximate Q-learning and discrete Q-Learning (especially) produce slower convergence. These experimental results show that the learning process can be accelerated to a significant degree if a priori knowledge of camera parameters and topology information about the environment is utilized, even though the state transitions, that is, the model of target movement in the scene, are not known in advance.

Because the node selection results may be affected by the use of different time intervals, we conducted the experiments for 50 trajectories using time intervals of 10–120 samples (in increments of 10), and set the exploration probability of the approximated Q-function to ε_t=0 (that is, there was no exploration in this test). The average visual score $AvgVisScore = \frac{1}{T}\sum_{t=1}^{T}\overline{M_{t+1}^{a_t}}$, total switching number $TotalSwitchNum = \sum_{t=1}^{T}\delta(a_t, a_{t+1})$, and *AvgSwitchNum* were then compared for Max, Game, POMDP, and the proposed method (ARL). For comparison, the visual scores of



(a) Average immediate reward (b) Average switch number per 10 selections

Fig. 5 Comparison between initialized Q-value and uninitialized Q-value



(a) Average visual score (b) Total switching number

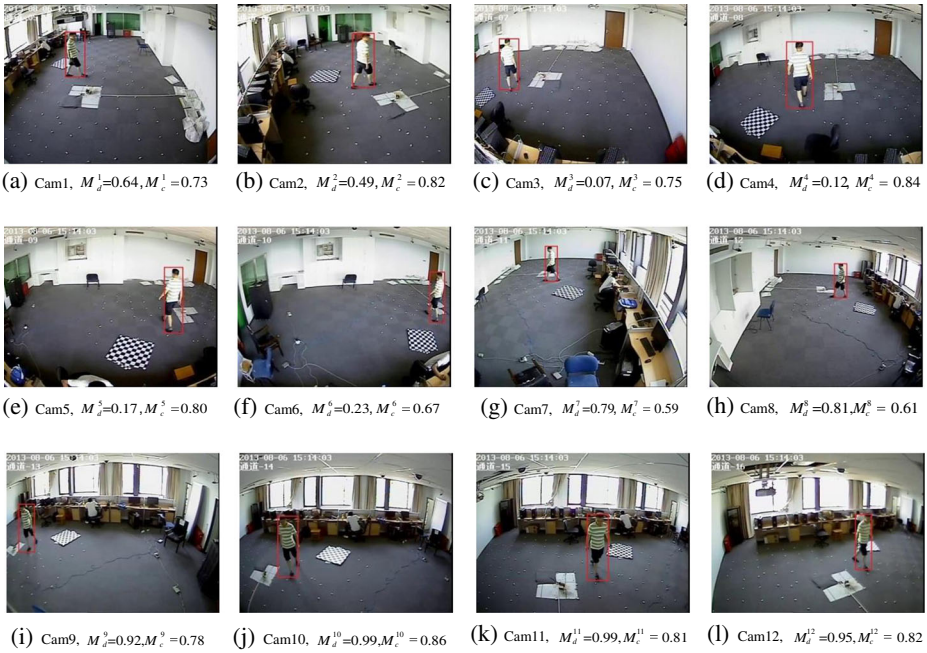
(c) Average switching number

Fig. 6 Simulation results under different time intervals

Max, Game, and POMDP were calculated using the proposed visual measurement. The resulting *AvgVisScore*, *TotalSwitchNum*, and *AvgSwitchNum* for the above methods are shown in Fig. 6. It is clear that the total switching number reduces as the time interval increases, whereas the average switching number per 10 selections increases and the average visual score tends to decrease with the reduced correlation between the current state and the next state. Further, the results of our approach are slightly superior to those given by the other three methods.

6.3 Real-life experimental results

We deployed 12 traditional calibrated non-smart cameras in an indoor scene. Natural video sequences were captured simultaneously by the cameras, and the visual processing for different video streams was conducted in parallel, thus simulating a distributed smart camera network. The action and learning modules of Q-learning in the control agent were also processed using multiple threads. A total of 120 video groups of 10 min in length were recorded as training data, and another five video groups of 60 min length were selected as test data. Neither exploration nor policy updating was performed during the test process. We found that the learning convergence during training was faster than for the simulations described in Section 6.2. This is because the target trajectories were more regular in the real scene than in the simulation environment, and a smaller state space was needed for training with respect to objects such as tables and chairs occupying certain spaces. The tracking algorithm itself is not the focus of this study; we used the VTD tracker [15] and manually revised the object location



(a) Cam1, M_d^1=0.64, M_c^1 =0.73 (b) Cam2, M_d^2=0.49, M_c^2 = 0.82 (c) Cam3, M_d^3=0.07, M_c^3 = 0.75 (d) Cam4, M_d^4 =0.12, M_c^4 = 0.84

(e) Cam5, M_d^5=0.17, M_c^5 =0.80 (f) Cam6, M_d^6=0.23, M_c^6 = 0.67 (g) Cam7, M_d^7=0.79, M_c^7 = 0.59 (h) Cam8, M_d^8 =0.81, M_c^8 = 0.61

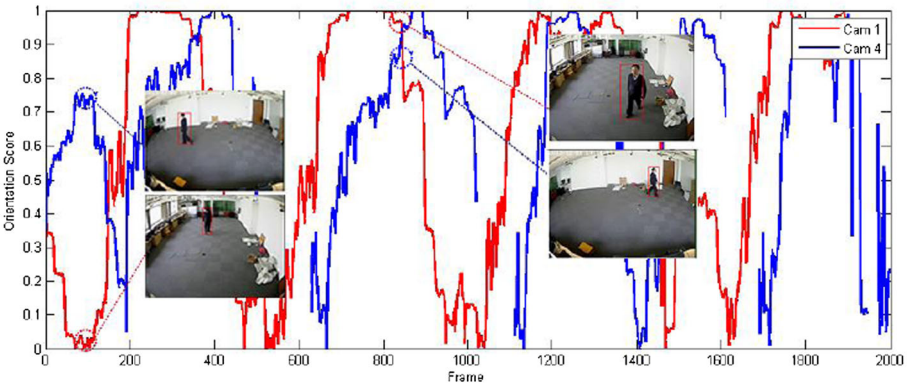
(i) Cam9, M_d^9=0.92, M_c^9 =0.78 (j) Cam10, M_d^{10}=0.99, M_c^{10} = 0.86 (k) Cam11, M_d^{11}=0.99, M_c^{11} = 0.81 (l) Cam12, M_d^{12} =0.95, M_c^{12} = 0.82

Fig. 7 Sample frames and tracking results for sequence 1

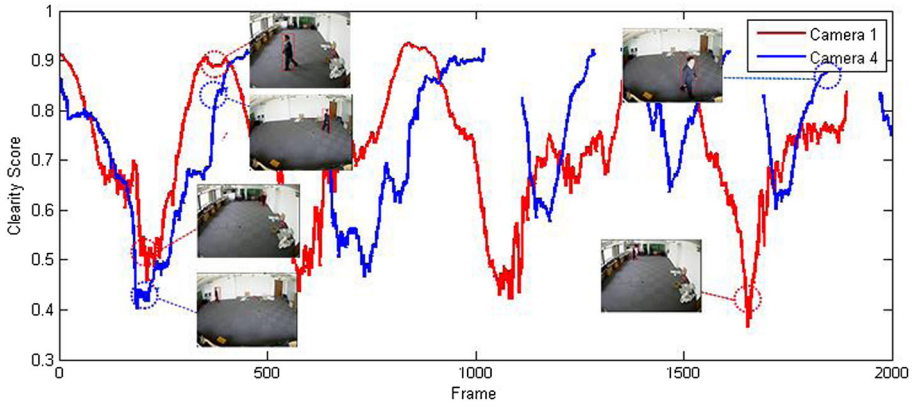
when it was in danger of being lost during tracking. The sampling image and target tracking results for video sequence 1 are shown in Fig. 7, with the orientation score M_d^i and clarity score M_c^i for camera i calculated using (36) and (37). The orientation and clarity scores are illustrated in the figures; the target has a higher orientation score when it moves directly towards the camera, and has a higher clarity score when it is closer to the camera.

The orientation, clarity, and total scores for camera 1 (Cam1) and camera 4 (Cam4) for frames 1–2000 of sequence 2 are shown in Fig. 8a–c. For the orientation curves, the angle in radians of the target relative to the x-axis of Cam1 and Cam4 in frames 90 and 830 is 0.09 and 2.65, respectively. The orientation scores of 0.007 and 0.75 in frame 90, and 0.97 and 0.84 in frame 830 better reflect the orientation of the target relative to that of the camera. Although the target reappears in the scene, some parts of the orientation curves (e.g., Cam4 near frames 1950 and 1350) are unstable, which causes varying degrees of jitter in the total score curves. This is mainly because the initial tracker performance is unstable, and some fitting errors are introduced when there are fewer fitting points. The resolution curves show images from frames 210, 380, 1650, and 1870. When the target is far away from the camera, the resolution of the target image is lower and the score is smaller. The resolution curves appear to have a certain amount of jitter in some areas (e.g., frame 1150 of Cam1), signifying errors in the tracking and calibration algorithm and a complex background.

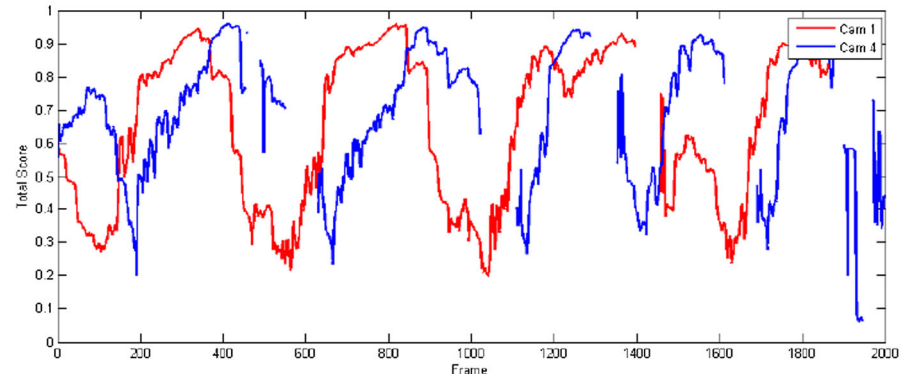
We compared our method (ARL) with Max, Game, and POMDP on five test video sequences. As in the simulation experiments, all the cameras were set to obtain tracking results, and the proposed visual evaluation metrics were used for comparison. Figure 9 shows the selection results for sequences 2 and 3, with a selection decision made every 20 s. In the figure, it can be seen that Max and Game make some false selections. This is because of the inconsistency between the visual score and the real target state in some views for some target



(a) Orientation score curves for Cam1 and Cam4



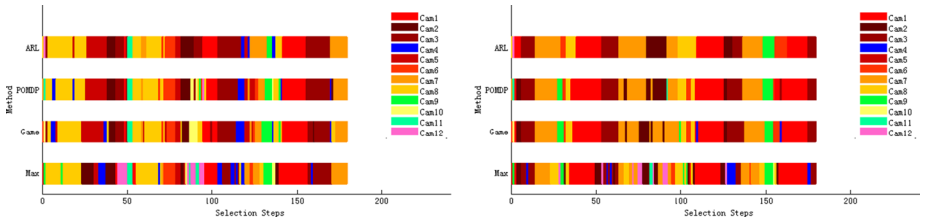
(b) Clarity score curves for Cam1 and Cam4



(c) Total visual information score curves for Cam1 and Cam4

Fig. 8 Visual information score curves for Cam1 and Cam4

errors. In particular, when the visual scores of multiple perspectives are similar or below a certain threshold, the Game selection result tends to produce jitter effects. For example, there is frequent switching in steps 65–75 of sequence 2 and steps 90–100 of sequence 3. In contrast,



(a) Results of camera selection for sequence 2

(b) Results of camera selection for sequence 3

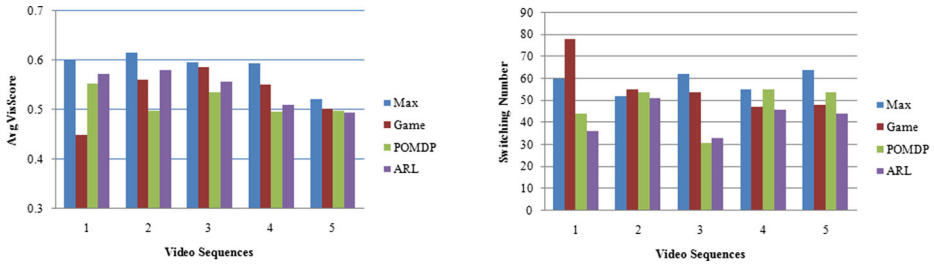
Fig. 9 Results of camera selection for sequences 2 and 3

our method effectively reduces the incidence of false selection and obtains stable results because of the Q-value learning from the training data.

The switching numbers and visual scores per step for five video sequences are shown in Fig. 10a and b. Greedy selection obtained a higher visual score per step for these sequences, although the scores for each method are generally similar. The proposed method has obvious advantages over the other methods in terms of the total switching number, especially greedy selection and Game.

7 Conclusion

As camera networks become extensively utilized in areas such as surveillance and human–computer interaction, finding the most informative and comfortable view from the mass of real-time video is of increasing importance. In this paper, we have proposed a dynamic node selection framework for surveillance applications in critical areas. To deal with the unknown state transitions of target motion and maximize the long-term reward, we trained a selection policy using reinforcement learning in the central controller. The chosen camera nodes extract effective visual features from their video streams and send the immediate reward to the central controller. To accelerate the learning process and reduce storage space, the target's current state and scene topology were used as a priori knowledge to initialize the Q-function, and an exploration strategy based on Gibbs' distribution was adopted to guarantee that the camera with the greatest Q-value had a greater exploration probability than under traditional random exploratory action selection. We utilized GMMs to represent the approximate Q-value function, and updated the GMM parameters sequentially via the mini-batch stepwise EM algorithm to meet the learning requirements of episodic sample updating. In addition, we measured the visual information and comfort of visual effects given by different cameras by integrating metrics on visibility, orientation, target



(a) Average visual scores per frame

(b) Total switching numbers

Fig. 10 Comparison of visual scores and switching numbers for the various methods

clarity, and the cost of switching cameras. Our study of single node selection and scheduling to maximize the expectation of visual information shows that good results can be achieved when fewer targets are in a partial overlap region. The implementation of the proposed approach on a camera network test-bed is planned for future work.

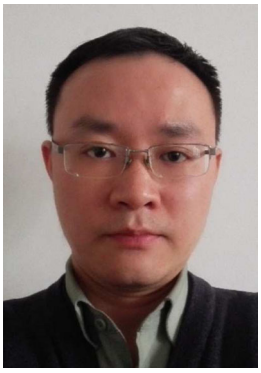
In terms of bandwidth consumption, because feature extraction and scoring are conducted on each camera node and the target association and fusion are distributed across all nodes, each camera need only send its current state and immediate reward to the central controller, and the central node transmits the selected camera's index to all nodes. The process of node selection is thus distributed, and the network bandwidth consumption is relatively small. However, a central controller is still needed, and the learning and approximation are centralized. Therefore, a decentralized selection policy is of interest, that is, cooperation and coordination between smart camera nodes should be considered.

Acknowledgments This work is supported by the National Natural Science Foundation of China (61272219, 61100110, 41305138, 61473310 and 61321491), Program for New Century Excellent Talents of Ministry of Education of China (NCET-04-0460), Science and Technology Plan of Jiangsu Province (BE2010072, BE2011058, BY2012190, and BY2013072-04), and Innovation Foundation of State Key Lab for Novel Software Technology of China (ZZKT2013A12).

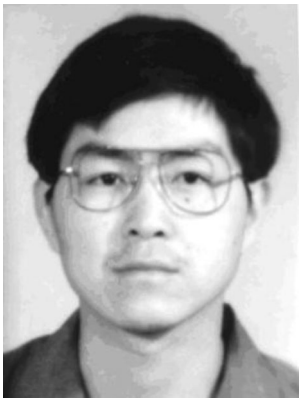
References

1. Agarwal A, Triggs B (2006) Recovering 3D human pose from monocular images. IEEE Trans Patt Anal Mach Intell 28(1):44–58. doi:10.1109/TPAMI.2006.21
2. Agostini A, Celaya E (2010) Reinforcement learning with a Gaussian mixture model. In Neural Networks (IJCNN), The 2010 Int Joint Conf : 1–8. doi: 10.1109/IJCNN.2010.5596306
3. Botvinick M, Niv Y, Barto A (2009) Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. Cognition 113(3):262–280. doi:10.1016/j.cognition.2008.08.011
4. Busoniu L, Babuska R, DeSchutter B, Ernst D (2010) Reinforcement learning and dynamic programming using function approximators. CRC press. doi: 10.1201/9781439821091
5. Charfi Y, Wakamiya N, Murata M (2009) Challenging issues in visual sensor networks. IEEE Wirel Commun 16(2):44. doi:10.1109/MWC.2009.4907559
6. Cohn D, Ghahramani Z, Jordan M (1996) Active learning with statistical models. J Artif Intell Res 4(1):129–145
7. Daniyal F, Taj M, Cavallaro A (2010) Content and task-based view selection from multiple video streams. Multimed Tools Applic 46:235–258. doi:10.1007/s11042-009-0355-z
8. Fu Y, Guo Y, Zhu Y (2010) Multi-view video summarization. IEEE Trans Multimed 12(7):717–729. doi:10.1109/TMM.2010.2052025
9. Gupta A, Mittal A, Davis L (2007) Cost: An approach for camera selection and multi-object inference ordering in dynamic scenes. Proceedings of IEEE International Conference on Computer Vision, Rio de Janeiro: 1–8. doi:10.1109/ICCV.2007.4408842
10. Han B, Joo S, Davis L (2011) Multi-camera tracking with adaptive resource allocation. Int J Comput Vis 91(1):45–58. doi:10.1007/s11263-010-0373-3
11. Huber M (2012) Optimal pruning for multi-step sensor scheduling. IEEE Trans Autom Control 57(5):1338–1343. doi:10.1109/TAC.2011.2175070
12. Javed O, Khan S, Rasheed Z (2000) Camera handoff: tracking in multiple uncalibrated stationary cameras. Proc Workshop Human Motion, IEEE: 113–118. doi: 10.1109/HUMO.2000.897380
13. Jiang H, Fels S, Little J (2008) Optimizing multiple object tracking and best view video synthesis. IEEE Trans Multimed 10(6):997–1012. doi:10.1109/TMM.2008.2001379
14. Kveton B, Theocharous G (2012) Kernel-based reinforcement learning on representative states. Association for the Advancement of Artificial Intelligence, pp 977–983
15. Kwon J, Lee K (2010) Visual tracking decomposition. IEEE Conf Comput Vision Patt Recognit: 1269–1276. doi: 10.1109/CVPR.2010.5539821

16. Li Y, Bhanu B (2011) Utility-based camera assignment in a video network: a game theoretic framework. IEEE Sensors J 11(3):676–687. doi:10.1109/JSEN.2010.2051148
17. Li Q, Sun Z, Chen S, Liu Y (2013) A method of camera selection based on partially observable Markov decision process model in camera networks. Ame Contrl Conf: 3833–3839. doi:10.1109/ACC.2013.6580424
18. Li W, Zhang W (2012) Sensor selection for improving accuracy of target localization in wireless visual sensor networks. IET Wireless Sens Syst 2(4):293–301. doi:10.1049/iet-wss.2012.0033
19. Liang P, Klein D (2009) Online EM for unsupervised models. In Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics. Assoc Comput Linguist: 611–619. doi: 10.3115/1620754.1620843
20. McLachlan G, Peel D (2004) Finite mixture models. New York: J. Wiley. doi:10.1002/0471721182
21. Mo Y, Ambrosino R, Sinopoli B (2011) Sensor selection strategies for state estimation in energy constrained wireless sensor networks. Automatica 47(7):1330–1338. doi:10.1016/j.automatica.2011.02.001
22. Monari E, Kroschel K (2009) A knowledge-based camera selection approach for object tracking in large sensor networks. Third ACM/IEEE Int Conf Distrib Smart Cam. doi:10.1109/ICDSC.2009.5289400
23. Park J, Bhat C, Kak A (2006) A look-up table based approach for solving the camera selection problem in large camera networks. In IEEE Workshop on Distributed Smart Cameras, Boulder, CO, USA, Oct. 31, 2006
24. Rudoy D, Zelnik-Manor L (2012) Viewpoint selection for human actions. Int J Comput Vis 97(3):243–254. doi:10.1007/s11263-011-0484-5
25. Rudoy D, Zelnik-Manor L (2012) Viewpoint selection for human actions. Int J Comput Vis 97(3):243–254. doi:10.1007/s11263-011-0484-5
26. Sato M, Ishii S (2000) On-line EM algorithm for the normalized Gaussian network. Neural Comput 12(2): 407–432. doi:10.1162/089976600300015853
27. Shen C, Zhang C, Fels S (2007) A multi-camera surveillance system that estimates quality-of-view measurement. Proc IEEE Int Conf Image Process, San Antonio: III 193-III 196. doi:10.1109/ICIP.2007.4379279
28. Singh S, Jaakkola T, Jordan M (1995) Reinforcement learning with soft state aggregation. Advances in neural information processing systems: 361–368. doi: 10.1162/089976600300015961
29. Soro S, Heinzelman W (2007) Camera selection in visual sensor networks. Proc IEEE Conf Adv Video Signal Based Surveill: 81–86. doi: 10.1109/AVSS.2007.4425290
30. Soro S, Heinzelman W (2009) A survey of visual sensor networks. Adv Multimed: 1–22. doi: 10.1155/2009/640386
31. Spaan M, Lima P (2009) A decision-theoretic approach to dynamic sensor selection in camera networks. In ICAPS: 1–8. doi: ocs/index.php/ICAPS/ICAPS09/paper/viewFile/758/1125
32. Sutton R, Barto A (1998) Reinforcement learning: an introduction. MIT Press. Cambridge, Massachusetts London, England. doi:10.1109/TNN.1998.712192
33. Tessens L, Morbee M, Lee H, Philips W, Aghajan H (2008) Principal view determination for camera selection in distributed smart camera networks. Sec ACM/IEEE Int Conf Distrib Smart Cam: 1–10. doi:10.1109/ICDSC.2008.4635699
34. Watkins C, Dayan P (1992) Q-learning. Mach Learn 8(3):279–292. doi:10.1007/BF00992698

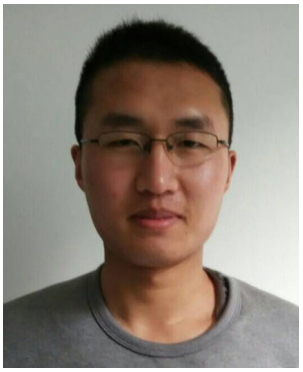


Qian Li is now a Ph.D. candidate in Computer Science and Technology Department of Nanjing University, China. He joined the State Key Laboratory for Novel Software Technology at Nanjing University in 2009. His research interests include Artificial Intelligence, Image and Video Process and Computer Aided Design.



Zhengxing Sun received the Ph.D. degree from NUAA in 1996 and finished his Post-doctoral researches in Nanjing University in 1999. He is now a full professor and academia committee man of the Department of Computer Science and Technology in Nanjing University. His research interests include Multimedia Computing, Computer Vision, and Perceptive Human-Computer Interaction.

Songle Chen is now a Ph.D. candidate in Computer Science and Technology Department of Nanjing University, China. He joined the State Key Laboratory for Novel Software Technology at Nanjing University in 2009. His research interests include Image and Video Process, Multimedia Analysis and Retrieval, Computer Graphics.



Shiming Xia is now a M.S. candidate in College of Meteorology and Oceanography of the PLA University of Science and Technology, China. He joined College of Meteorology and Oceanography of the PLA University of Science and Technology in 2013. His research interests include Image and Video Process, Machine Learning.