Multiple Kernelized Correlation Filters (MKCF) for Extended Object Tracking Using X-band Marine Radar Data

Yi Zhou, Tian Wang, Ronghua Hu, Hang Su, Yi Liu, Xiaoming Liu, Jidong Suo, and Hichem Snoussi

Abstract

Conventional tracking filter in marine radar only concerns the position or shape of the object. When the target encounters the clutter interruption or ship occlusion, the tracker is easy to drift away due to the wrong target association. Since in short-range maritime surveillance, X-band marine radar captures the object in extended region with varying intensities, in this paper, combining position, shape, and appearance of the target together, multiple kernelized correlation filters (MKCF) are proposed to conduct a single object tracking in the real marine radar. By automatic initializing KCF on a target in different time steps and fusing these multiple KCFs via the maximum likelihood, the proposed tracker implicitly uses multiple instances of the target to improve the robustness of the long-term tracking. Bounding rectangles of the multiple trackers also enhance the reliability of ship segmentation via a voting procedure. In the real ship tracking experiment, the proposed tracker performs favorably against the conventional radar tracker and top-ranked visual tracker in the cases of clutter interruption, ship occlusions, and scale changing. We believe that our proposal sheds light on that the intensity distribution of the extended object could be valuable for target association in marine radar data. To encourage the further researching, our tracking framework is made open-source.

Yi Zhou, Yi Liu, Xiaoming Liu, Jidong Suo are with the Department of Electronic Information Engineering, Dalian Maritime University, Dalian, 116026, China. Email: ({yi.zhou, lydmu, lxmdmu, sjddmu}@dlmu.edu.cn).

Tian Wang is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, 100191, China. Email:(wangtian@buaa.edu.cn).

Ronghua Hu and Hichem Snoussi is with ICD-LM2S, University of Technology of Troyes, Troyes, 10004, France. Email:({ronghua.hu, hichem.snoussi}@utt.fr)

Hang Su is with the Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China . Email:(suhangss@tsinghua.edu.cn).

Summary of Notation

MKCF	the proposed Multiple Kernelized Correlation Filters.
T_i	the ith component KCF tracker in the MKCF.
γ	a real value in the range axis of the *range-azimuth* coordinate.
θ	a real value in the azimuth axis of the *range-azimuth* coordinate.
$\boldsymbol{\upsilon}$	location vector in the *range-azimuth* coordinate, $\boldsymbol{\upsilon} = [\gamma, \theta]$.
$\boldsymbol{\delta}$	offset vector of the translation in the *range-azimuth* coordinate.
w	width of the rectangle.
h	height of the rectangle.
$\boldsymbol{b}$	rectangle vector $\boldsymbol{b} = [\gamma, \theta, w, h]^T$ with center coordinate, width and height.
$\boldsymbol{b}_0$	initial rectangle vector for initializing a KCF tracker.
$\boldsymbol{b}_i$	a rectangle vector for one generated sample.
$\tilde{\boldsymbol{b}}$	estimated rectangle vector of a single KCF.
$\tilde{\boldsymbol{b}}_i$	estimated rectangle vector of the ith component KCF in MKCF.
$\tilde{\boldsymbol{b}}_{in}$	estimated rectangle vector of the ith component KCF in MKCF at nth time step.
$\hat{\boldsymbol{b}}$	estimated rectangle vector of the MKCF.
$\hat{\boldsymbol{b}}^-$	estimated rectangle vector of the MKCF in previous time step.
$\hat{\boldsymbol{b}}_n$	estimated rectangle vector of the MKCF at time step n.
$\mathbf{r}$	selected bounding rectangle vector for initializing a new KCF tracker.
$\mathbf{r}_n$	selected bounding rectangle vector at time step n.
$\mathbf{Y}$	label matrix for training the classifier in KCF.
y_i	the ith element in the label matrix.
$\mathbf{Y}'$	response matrix of the testing in KCF.
y'_i	the ith element in the response matrix $\mathbf{Y}'$.
y'_p	the maximum value of the response matrix $\mathbf{Y}'$.
$\mathbf{Y}'_i$	response matrix of the ith component KCF in MKCF.
y'_{pi}	the maximum value of the response matrix of the $\mathbf{Y}'_i$.
$\mathbf{x}$	the echo intensities matrix of the target. It is used as the reference of the KCF.
$\mathbf{x}_i$	the ith circularly shifted matrix of the reference based on $\mathbf{x}$.
$\mathbf{z}$	the echo intensities matrix of the observation. It is used as the observation of the KCF.

$\mathbf{z}_i$	the ith circularly shifted matrix of the observation based on $\mathbf{z}$.
s	score of the Peak to Side-lobe Ratio (PSR).
s_i	score of the PSR of the ith component KCF in MKCF.
μ_s	the mean of the side-lobe region.
σ_s	the standard deviation of the side-lobe region.
λ	the regular coefficient of the classifier.
σ_k	the standard deviation of the isotropic Gaussian kernel.
N_{th}	the allowed number of the fused trackers.
N_{max}	the upper bound of the number of the fused trackers.
N_{min}	the lower bound of the number of the fused trackers.
S_{th}	PSR threshold controls the updating KCF, deleting KCF and adaptively changing N_{th}.
O_{th}	Overlapping ratio threshold controls the initializing the new tracker given the selected $\mathbf{r}$.
$f(\cdot)$	regression function of the classifier, which maps the input feature to a real value in the range of $[0, 1]$.
$\varphi(\cdot)$	implicit function maps the input feature to higher dimensional feature.
$\kappa(\cdot)$	kernel function for the element of the Kernel matrix.
$\boldsymbol{I}$	the identity matrix.
$\boldsymbol{I}_2$	the identity matrix with size 2×2.
$\boldsymbol{K}$	kernel matrix.
$\boldsymbol{\alpha}$	parameter matrix of the classifier function in KCF.
$\boldsymbol{K_{xx}}$	the reference-self kernel matrix between the all the circularly shifted $\{\mathbf{x}_i\}$ references and the original $\mathbf{x}$.
$\boldsymbol{K_{zz}}$	the observation-self kernel matrix between the all the circularly shifted $\{\mathbf{z}_i\}$ observations and the original $\mathbf{z}$.
$\boldsymbol{K_{xz}}$	the mutual kernel matrix between the all the circularly shifted $\{\mathbf{x}_i\}$ references and the original observation $\mathbf{Z}$.
$\mathcal{F}(\cdot)$	the Fourier transform.
$\mathcal{F}^{-1}(\cdot)$	the inverse Fourier transform.

Contents

Multiple Kernelized Correlation Filters (MKCF) for Extended Object Tracking Using X-band Marine Radar Data

I. Introduction

X-band marine radar is widely exploited for maritime surveillance. In the station of vessel traffic service (VTS) center, ship tracking of marine radar reflects the traffic dynamics in all weather conditions. It plays an important role in collision avoidance, search and rescue. Traditional tracking in marine radar is based on the point target hypothesis, i.e. one target is represented by a point (one single resolution cell in radar processing units) per frame. However, marine radar is a high-resolution sensor. In each scan, a target can be found in a group of points (see Figure 1). In order to feed only one point to the traditional tracking algorithm, clustering and centroid extraction are conducted in the pre-tracking stage. Consequently, the shape and appearance of the target are ignored. This assumption is reasonable for the large-range surveillance, in which scenario, echo intensity reflected by the target is far stronger than the background clutter and the clustered region of the object is often isolated. While for the short-range surveillance, the echo region of one object is composed of many measurements scattered from spatially different sources on the same object. To track a target with a group of points, extended object tracking (EOT) [1]–[4] or extended target tracking (ETT) [5], [6] are proposed to handle the high-resolution marine radar data [7]–[9]. The objective of EOT is to estimate both the centroid and shape of the target object. The group of points is fitted to the defined simple shapes such as rectangle or ellipse. In real marine radar application, point or shape based tracking filter has difficulty to deal with the clutter interruption and ship occlusion. In those cases, the region of the target is easily mixed with the strong background clutter or occluded by neighboring objects. Trajectory or shape fitting alone cannot be able to locate the target's position accurately. It causes wrong target association in radar tracking and results in tracker drifting. Therefore, the radar operators of VTS need to monitor the radar tracking in 24 hours in turns, and often has to correct the wrong tracker manually especially in those complex electromagnetic environment.

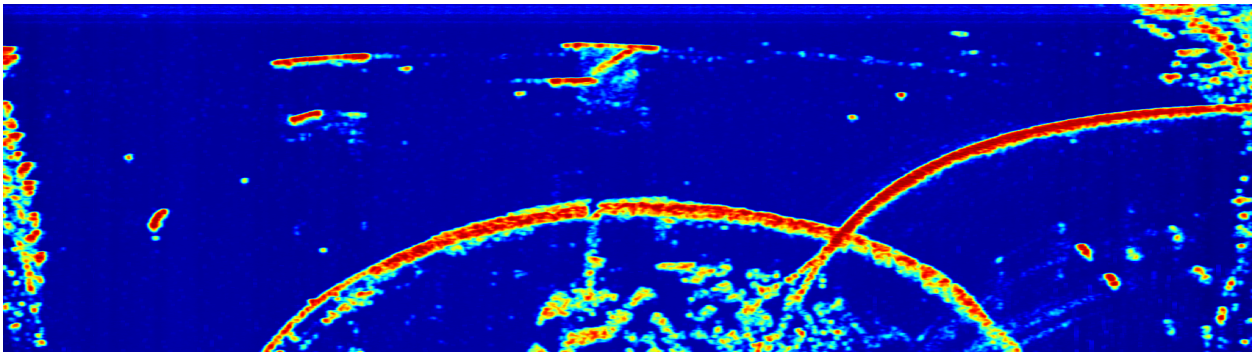


Fig. 1: Heat map of the radar image captured in an inner river site. Ships, buoys in the river, bank of the river and railway bridge across the river show strong intensities in deep red color. When the ship is approaching the radar site (top center of the image), multi-path clutter makes the shape of the ship irregular and causes interruption to the neighbor ship. The radar image is in *range-azimuth* format (the vertical axis maps the *range* units, and the horizontal denotes the *azimuth*).

In this paper, we propose to use position, shape and appearance of the object together for radar tracking in a scene of short-range inner river surveillance. When the intensities of the echo from the target is modeled as appearance feature, we enhance the tracker with more capability to recognize the same object in the case of external interruption. This approach is related to the visual tracking in computer vision, in which the visual object consists of pixels with varied illumination strength.

Visual tracking is inspired by human visual instincts. Humans not only track the target based on the motion model (the speed and direction of the movement), but also remember what the target looks like. In early visual tracking, the appearance model is encoded as contour [10], histogram [11], etc. to represent the looks of the target. These features encode the shape, texture or luminance information of the target appearance. Recently, the convolutional neural networks (CNNs) [12] are pre-trained using raw pixels with a fixed input size on a large amount of labeled image dataset [13]. It has been observed that the features extracted from the CNNs are generic and can be exploited for visual applications [14]–[16]. To tolerate the changes of appearance frame by frame, the appearance model is learned online during the tracking process. That is known as tracking-by-detection in many visual applications [17], [18].

The computation of tracking is subject to the real-time constraint both in radar and computer

vision. Among the wide range of visual tracking algorithms, the kernelized correlation filter (KCF) [19] achieves the appealing results both in accuracy and time consumption by taking only the intensity of raw pixel as feature. Many correlation filter based visual tracker have ranked the top place in recent visual tracking benchmark [14], [20]–[23]. However, most of them employ the appearance template with the fixed size and update it in a running average, which are not able to handle the multiple models or scales of the target in a long-term tracking. In this paper, we propose to use multiple KCFs (MKCF) to track a single radar object with varying appearance in a complex electromagnetic environment.

The main contributions of this paper are summarized as follows:

1. Design a novel visual tracking framework for ship tracking in the marine radar. Compared to the point or shape based radar trackers, the proposed MKCF utilizes the multiple instances of the intensity of the target echo to identify the target. It improves the the reliability of target association in the case of external interruption.

2. Integrate the short-term KCF trackers in different time steps and fuse them in the maximum likelihood criterion. The proposed MKCF extends KCF tracker with the capability of handling scale changes and multiple instances learning, and obtains an appealing performance gain in the long-term tracking both in accuracy and robustness.

3.The tracking results of the tracker and object segmentation are combined together for automatic initializing new tracker. Tracking helps to locate the target when the segmentation is confused by the connected echo region in the cases of ship occlusion and clutter interruption. In turn, the segmentation helps the tracker to capture the scale or appearance changing by initializing new tracker with the qualified blob.

II. Related Works

In our work, we explore a typical visual tracker to do object tracking in the radar signal. We also compare it with the traditional radar trackers. So in this section we give a short overview on the two related domains.

A. Object tracking in marine radar

In maritime surveillance, the position and velocity of vessel are measured at discrete time instants via the track-while-scan X-band radar sensor. Majority vessels travel in relatively low

speed with little maneuvering [24]. Tracking filter is used to smooth the measured data and form the tracks during the scans. Widely popular Kalman filter provides an optimal estimation for linear system in which the system and measurement error are assumed Gaussian noise [25]. The alpha-beta filter [26] is a simpler derivation of the Kalman filter that was successfully used to estimate a moving target with stable velocity.

Conventional tracking filter makes the hypothesis of one point per target [27], [28]. However, point representing often causes associating problems in the cases of closely spaced targets and noisy measurements [29]. To properly track the extended targets, ETT or EOT is proposed [2]–[6]. Contrast with point-based tracking filter, EOT deals with not only the kinematic state of the object but also its extension (width and length). Characterizing the ellipsoidal extensions by a random matrix, Koch [2] develops a popular EOT framework in Bayesian approach. Taking the measuring noise into account, Feldmann [3] improves the performance of EOT in [2]. In [6], [8] probability hypothesis density (PHD) filter is enhanced in order to deal with the case of multiple extended targets.

Recently, EOT based approaches are exploited on the real marine radar [7]–[9]. In [7], combining the particle filtering and track before detect technique, target states and geometrical shape are estimated frame by frame. In a recent EOT review paper [30], the Gamma Gaussian inverse Wishart (GGIW) density is highlighted to model the single extended target state. It is noticed that GGIW based EOT model performs favorably in the tracking application of X-band radar [8]. Using same radar data, [9] proposes a signal processing chain composed by a pixel-level detector and a joint probabilistic data association (JPDA) tracker to do the multiple ETT. These methods apply the EOT into the conventional multiple target tracking framework. The experimental radar data contains sparsely distributed ships with moderate clutter interruption.

In this work, rather than define a motion model, we exploit feature learning-based visual tracker for a single radar object tracking. We testify that the intensity distribution of the extended object helps to enhance the robustness of the target tracking in marine radar.

B. Visual tracking

Early visual tracking has a tight link with radar object tracking. She borrows the idea of Kalman filter based point tracking [31]. Estimating and updating steps are combined to track the visual object in the noisy environment. Latter particle filter [10] is proposed to handle the non-

linear and non-Gaussian situations. It improves the robustness. But in real visual applications, visual tracker still suffers to the illumination changing, scale variation, object rotation. During the implementation of particle filter for visual tracking, researchers find that the likelihood measurement plays more important role than the dynamic model. With only sophisticated feature and accompanying similarity measurement, the tracker would have better results even without the motion model [32].

In order to handle the changes of appearance feature during the tracking, feature learning based classifiers are designed to distinguish the target from the surrounding environment [17], [18], [33], [34]. Tracking is equivalent to detecting the same object in the successive frames. In recent year, the typical KCF tracker outperforms those far more complicated trackers such as Struck [34] or TLD [18] on the open video benchmarks, running at amazing 300 frames-per-second [35]. Correlation filter becomes the baseline for many vision trackers, especially for the applications with the real-time requirement. However, the fixed template size of KCF can not handle the scale changes of object. Danelljan [20] proposes to learn discriminative correlation filters from a scale pyramid model. Bibi [36] fuses the scale variation into the learning procedures. Scale Adaptive Kernel Correlation Filter (ASKCF) in [21] gives a multiple scales searching strategy. By exhaustive searching the candidate scales, multi-kernel based tracker achieves comparable performance when the tracked object is not moving fast. Multi-kernel learning based KCF is first introduced in [37] and upgraded in [23] to enhance the discriminability of the visual feature, where two kernels in two complementary feature spaces are learned together.

To further improve the performance of correlation filter based tracking, the hand-crafted [20] and CNNs-generated features [14], [15] are proposed. Although more discriminative features bring better classifying performance, it cost heavier computing time. By contrast, without heavy feature computing, in this paper we integrate multiple correlation filters with only the simple raw pixel as feature to gain a competitive tracking performance.

III. Preliminary: Visual Tracking with the Kernelized Correlation Filter

This is a preliminary section on introducing KCF before the proposed MKCF. First we illustrates the general framework of the tracking-by-detection. In the following, we shows how

the circular structure of the dense samples speed up training and testing in KCF. At the end, the naive running average for updating in KCF is described.

A. Framework of the tracking-by-detection tracker

The common task for visual tracking is to initialize a target in the first frame with a bounding box and then locate the box on the target of interest accurately in the subsequent frames. To track arbitrary objects, machine learning techniques are exploited to learn the discriminant classifier online. Tracking is equal to detecting the learned discriminative features, which is called Tracking-by-detection [17], [18], [33]–[35]. It runs in a loop of training, testing, and online updating (see Figure 2).

In the initial frame, known the location $\boldsymbol{b}_0$ (bounding rectangle $[\gamma_0, \theta_0, w_0, h_0]^T$, including center (γ_0, θ_0) in *range-azimuth* coordinate, width w_0, and height h_0) of the target, positive samples close to the given center and negative samples beyond a close distance are generated as a position set $\{\boldsymbol{b}_i\}_{i=1}^m$. Here $m \in \mathbb{N}$ denotes the total number of samples. On each sampling location, appearance feature $\mathbf{x}_i$ is computed (in this paper, we use the raw intensities in a rectangle region of the radar image as feature, therefore $\mathbf{x}_i \in \mathbb{R}^d$, $d = w_0 \times h_0$). Based on the distance of the sampled position $\boldsymbol{b}_i$ to $\boldsymbol{b}_0$, a label value y_i, ranging from 1 to 0, is associated with corresponding feature $\mathbf{x}_i$. If the label y_i is near to 1, it means that the sampling position $\boldsymbol{b}_i$ is close the true target and the input feature $\mathbf{x}_i$ is more like from the target. Otherwise, if the label is close to 0, it indicates that the sampling position is far away to the target center and the feature is taken from the background. Given m feature-label pairs $\{\mathbf{x}_i, y_i\}_{i=1}^m$, a classifier function $f(\cdot)$ is trained to map a feature to a class label, i.e. $y_i = f(\mathbf{x}_i)$.

In latter frame, classifier function $f(\cdot)$ is used to test the features $\{\mathbf{z}_i\}_{i=1}^m$ from the candidate positions $\{\boldsymbol{b}_i^-\}_{i=1}^m$, which are sampled around the previous location $\hat{\boldsymbol{b}}^-$. The location $\hat{\boldsymbol{b}}$, where the tested feature has the maximum label value, is chosen as the target's position in that frame.

Finally, newly computed feature $\mathbf{x}_i'$ around the new position $\hat{\boldsymbol{b}}$ and defined labels y_i are used to online update the classifier function. It is noticed that when a KCF tracker is initiated with the bounding box $\boldsymbol{b}_0$, it keeps the same width and height of the target for training, testing and updating. The label set in the updating phase reuses the same $\{y_i\}_{i=1}^m$ in the training phase, assuming that the inferred $\hat{\boldsymbol{b}}$ is accurately holding the true target. In fact this is a high risk for

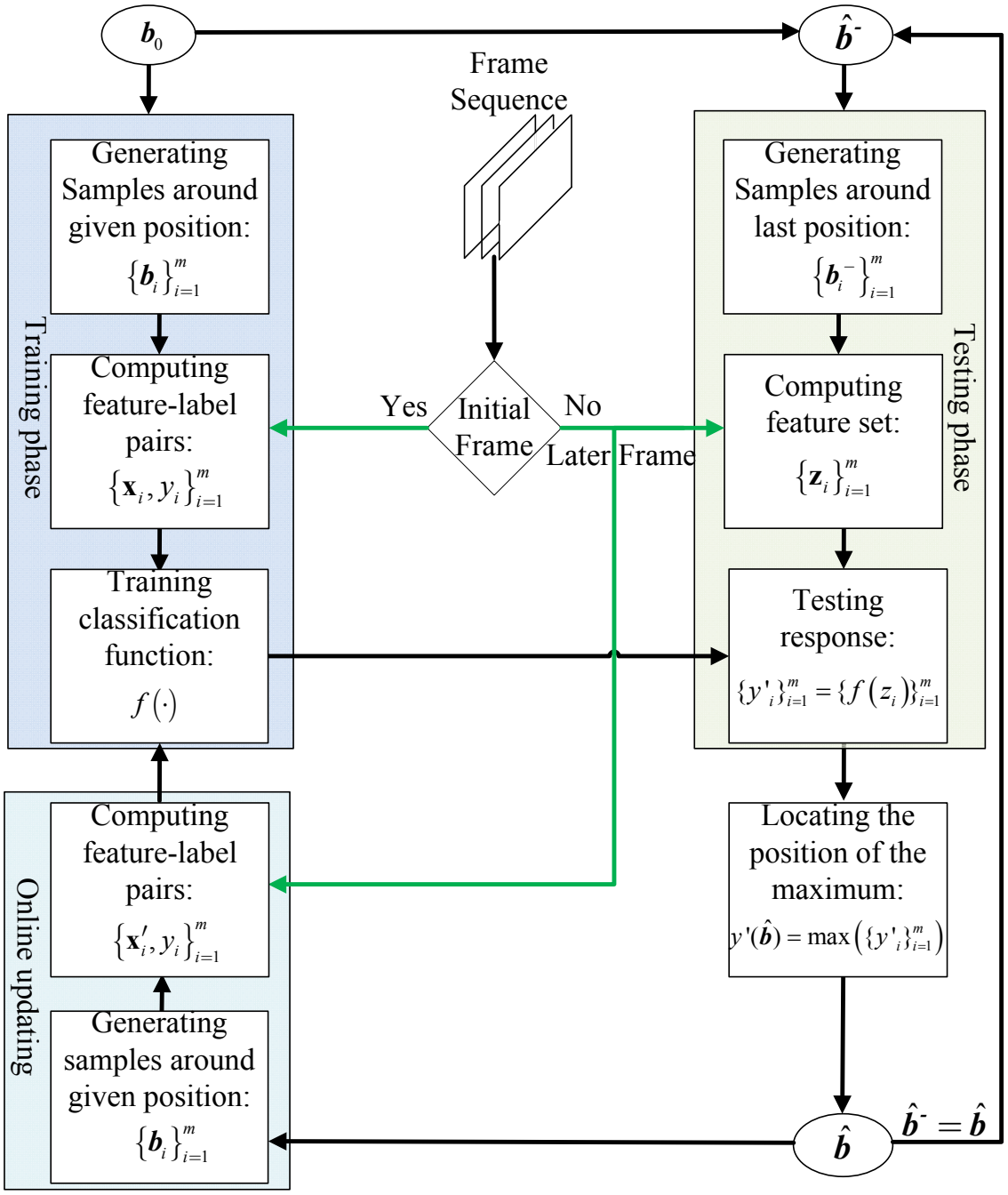


Fig. 2: General framework of the tracking-by-detection tracker. With initial frame and given location $\boldsymbol{b}_0$, classifier function $f(\cdot)$ is trained in the training phase. In latter frame, around the previous estimation $\hat{\boldsymbol{b}}^-$, testing phase generates the response of classification $\boldsymbol{y}'$. The location $\hat{\boldsymbol{b}}$ of the maximum response y' estimates the new position of the target. New feature-label pairs around $\hat{\boldsymbol{b}}$ are used to update the classifier function.

KCF, a little error in estimation would bring wrong labels to the samples. In consequence, the polluted samples would degrade the classifier function $f(\cdot)$ and lead to track loss.

B. Regularized least squares with kernel

Given the training feature-label pairs $\{\boldsymbol{x}_i, y_i\}_{i=1}^m$, the learning objective is to infer the parameters of the classifier function which minimize the regularized risk [38]. In the linear regression case, function $f(\cdot)$ is defined as a linear form in Eq. (1). The goal of training is to find the coefficients $\boldsymbol{w} \in \mathbb{R}^d$ (d equals to the dimensions of the feature $\boldsymbol{x}$), which minimizes the squared error over all sampling $\boldsymbol{x}_i$ and their regression value y_i. Eq. (2) defines the objective of optimization by minimizing the quadratic loss with regularized coefficient λ. This approach is also called the Regularized Least-Squares Classification (RLSC) in [39].

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i, \tag{1}$$

$$\min_{\boldsymbol{w}} \sum\nolimits_{i=1}^{m} \left(f(\boldsymbol{x}_i) - y_i\right)^2 + \lambda \left\|\boldsymbol{w}\right\|^2. \tag{2}$$

It is known that improving the dimensions of the input feature $\boldsymbol{x}$, the simple linear classification function $f(\cdot)$ could gain a better performance. Kernel based method [40], maps the input $\boldsymbol{x}$ to a new space $\varphi(\boldsymbol{x})$ with higher dimensions. When put the $\varphi(\boldsymbol{x})$ in the optimization process, the detailed form of function $\varphi(\cdot)$ is not necessary, only the inner product of $\varphi(\cdot)$ called kernel function κ in Eq. (3) is needed,

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left\langle \varphi(\boldsymbol{x}_i), \varphi(\boldsymbol{x}_j) \right\rangle. \tag{3}$$

If view the $\varphi(\boldsymbol{x})$ as the new feature space and defines $\boldsymbol{w}$ as follows:

$$\boldsymbol{w} = \sum\nolimits_{i=1}^{m} a_i \varphi(\boldsymbol{x}_i). \tag{4}$$

Putting the kernel function of Eq. (3) and weights Eq. (4) into the Eq. (2), the new objective optimization is defined as the Eq. (5):

$$\min_{\alpha} \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{m} a_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \right)^2 + \lambda \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{5}$$

Obviously, the objective function does not need the mapping function $\varphi(\cdot)$ any more. The KCF uses the Gaussian kernel by

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\left(-\frac{1}{\sigma_k^2} \left(\left\|\boldsymbol{x}_i\right\|^2 + \left\|\boldsymbol{x}_j\right\|^2 - 2\boldsymbol{x}_i^T \boldsymbol{x}_j \right) \right). \tag{6}$$

Here, σ_k is the the standard deviation of the isotropic Gaussian kernel.

The solution of Eq. (5) is given by (see [39])

$$\boldsymbol{a} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \boldsymbol{y}. \tag{7}$$

Here $\boldsymbol{a}$ =$[a_1, ..., a_m]^T$, $\boldsymbol{y} = [y_1, ..., y_m]^T$, $\mathbf{I}$ is the identity matrix. $\mathbf{K}$ denotes kernel matrix with $m \times m$ dimensions. In matrix $\mathbf{K}$, one element $\mathbf{K}_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

In the testing phase, the probability of a new input $\mathbf{z}$ being from the target feature is computed by

$$y' = f(\varphi(\boldsymbol{z})) = \left\langle \sum_{i=1}^{m} a_i \varphi(\boldsymbol{x}_i), \varphi(\boldsymbol{z}) \right\rangle = \sum_{i=1}^{m} a_i \kappa(\boldsymbol{x}_i, \boldsymbol{z}). \tag{8}$$

C. KCF employs circular structure for fast training and testing

Observing Eq. (7), computing $\mathbf{K}$ needs to calculate all the kernel function in every sample pair. By circular shifting the reference template $\mathbf{x}$ (a matrix with size $w \times h$), KCF gets the special dense samples, which forms kernel matrix with circular structure. In other words, **samples are densely sampled in every point of the given rectangle**. All the labels $\{y_i\}_{i=1}^{m}$ for these samples compose a label matrix $\mathbf{Y}$ with $w \times h$ dimensions, which is a needle-shaped 2d Gaussian with covariance $\sigma_y^2 \boldsymbol{I}_2$ in that rectangle, here σ_y is a small standard deviation for the isotropic Gaussian, $\boldsymbol{I}_2$ means the identity matrix with size 2×2. **The needle-shaped Gaussian label matrix means that only a few pixels around the center have the possibility to be chosen as the positive position for training.** Eq. (7) could be viewed as the matrix multiplication between the inverse of a circular matrix and label vector. This multiplication is in fact a convolution between one row of circular matrix and one label vector in $\mathbf{Y}$. As described in Convolution Theorem, the convolution in space domain corresponds to an element-wise multiplication in Fourier domain. So, the computing of the matrix $\boldsymbol{\alpha}$ can be expedited by the Fast Fourier Transform (FFT) (see [19]).

$$\boldsymbol{\alpha} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{Y})}{\mathcal{F}^*(\mathbf{K}_{xx}) + \lambda}\right). \tag{9}$$

Symbol $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and the inverse respectively, $\mathcal{F}^*$ is the complex conjugation of $\mathcal{F}$. Here the division in Frequency domain is computed element-wise. It is worth noting that in this paper all the divisions and multiplications in Frequency domain are computed element-wise. $\mathbf{K}_{xx}$ represents the kernel matrix. The subscript xx indicates that

kernel function takes the sample $\mathbf{x}$ and $\mathbf{x}$'s own circular shifting mode as input. For a Gaussian kernel, $\mathbf{K}_{xx}$ is defined by

$$\mathbf{K}_{xx} = exp\left(-\frac{1}{\sigma_k^2}\left(\|\mathbf{x}\|^2 + \|\mathbf{x}\|^2 - 2\mathcal{F}^{-1}\left(\mathcal{F}^*\left(\mathbf{x}\right)\mathcal{F}\left(\mathbf{x}\right)\right)\right)\right). \tag{10}$$

Similarly, the response of testing can be written by

$$\mathbf{Y}' = \mathcal{F}^{-1}\left(\mathcal{F}^*\left(\mathbf{K}_{xz}\right)\mathcal{F}\left(\boldsymbol{\alpha}\right)\right), \tag{11}$$

where

$$\mathbf{K}_{xz} = exp\left(-\frac{1}{\sigma_k^2}\left(\|\mathbf{x}\|^2 + \|\mathbf{z}\|^2 - 2\mathcal{F}^{-1}\left(\mathcal{F}^*\left(\mathbf{x}\right)\mathcal{F}\left(\mathbf{z}\right)\right)\right)\right). \tag{12}$$

The location of the peak value y'_p in the response $\mathbf{Y}'$ indicates the target's position.

D. Updating with naive running average in KCF

Correlation filter based tracker is vulnerable to appearance changing. To keep the continuous tracking, KCF uses a simple updating scheme based on the running average on both the referred feature $\mathbf{x}$ and the classifier's key parameter $\boldsymbol{\alpha}$. In each time step t, once the candidate $\mathbf{z}_t$ is chosen as the target, the reference feature $\mathbf{x}_t$ in time t is updated by the combination of the new feature $\mathbf{z}_t$ and the previous feature $\mathbf{x}_{t-1}$ with a learning rate η (see Eq.(13)). At the same time, $\mathbf{z}_t$ is viewed as the reference feature $\mathbf{x}$ in Eq. (10) to compute $\boldsymbol{k}_{z_t z_t}$. Replace $\boldsymbol{k}_{xx}$ in Eq. (9) with $\boldsymbol{k}_{z_t z_t}$, the key parameter $\boldsymbol{\alpha}_{z_t}$ of the new classifier based on $\mathbf{z}_t$ is trained. Similarly, the newly trained $\boldsymbol{\alpha}_{z_t}$ is combined with the previous $\boldsymbol{\alpha}_{t-1}$ with the same learning rate η to compute the classifier's parameter $\boldsymbol{\alpha}_t$ in time step t (see Eq. (14)).

$$\mathbf{x}_t = \mathbf{z}_t * \eta + (1-\eta) * \mathbf{x}_{t-1}. \tag{13}$$

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{z_t} * \eta + (1-\eta) * \boldsymbol{\alpha}_{t-1}. \tag{14}$$

As we have discussed, KCF is a tracking-by-detection tracker. The classifier parameter $\boldsymbol{\alpha}_t$ is very critical to the tracking performance. The running average of updating the reference $\mathbf{x}_t$ and $\boldsymbol{\alpha}_t$ is simple, but it is error prone and severely affects the performance of KCF in the long-term tracking. In fact the running average is based on two assumptions:

(1) The feature $\mathbf{z}_t$ and parameter of the classifier $\boldsymbol{\alpha}_t$ are linearly varied.

(2) The estimation of KCF in time t is precise and feature $\mathbf{z}_t$ is extracted from the true target.

However the two prerequisites are not consistently satisfied in visual tracking.

(1) The feature of the target often varies in the non-linear form, when it caused by the self-rotation, scale varying or object occlusion. Running average of Eq.(13) spoils the discrimination of the new feature $\mathbf{z}_t$, since it may has no linear connection with previous reference feature $\mathbf{x}_{t-1}$. In addition, any changes in the reference feature $\mathbf{x}_t$ even in linear form would cause nonlinear changing in $\boldsymbol{\alpha}_t$, according the nonlinear training Eq. (9) and kernel Eq. (10). Therefore the naive running average in Eq. (14) may work in the short term when the reference feature is stable. However it is error-prone in the long term tracking, since the visual feature often undergoes the varying.

(2)The varying feature of the target-self and the occlusion from other objects would degrade the tracking performance. If the inaccurate feature $\mathbf{z}_t$ is used to update the reference feature $\mathbf{x}_t$ and the parameter $\boldsymbol{\alpha}_t$ of the classifier, the errors would be accumulated frame by frame. The tracker would gradually drift away and lost the target totally.

IV. Proposed MKCF

In this section, we firstly analyze the KCF in the perspective of maximum likelihood. Next, we discuss how to monitor the status of a single KCF tracker and the idea of transition from relay tracking to the proposed MKCF. Then we propose to fuse the multiple KCF trackers via maximum likelihood. For automatic initializing the new tracker, in following subsection, object segmentation and blob selecting method are discussed. At the end of this section, the proposed MKCF is summarized.

A. Understanding KCF from the Maximum Likelihood Perspective

As it was mentioned before, KCF circularly shifts the target patch $\mathbf{x}$ (with w pixels in width and h pixels in height) to generate dense samples $\{\mathbf{x}_i\}_{i=1}^m$ ($m = w \times h$) in the reference frame. Then the pre-defined labels $\{y_i\}_{i=1}^m$ (ranging from 0 to 1) mark the probability that $\mathbf{x}_i$ is the target for each sample. Since all the samples are circular shifting based on $\mathbf{x}$, the m labels of $\{y_i\}_{i=1}^m$ can be aligned in a needle-shaped 2d Gaussian $\mathbf{Y}$ which has the same size as $\mathbf{x}$. In the training phase, the pre-defined labels $\{y_i\}_{i=1}^m$ and kernel $\mathbf{K}_{xx}$ between $\mathbf{x}$ and $\{\mathbf{x}_i\}_{i=1}^m$ are used to train a model $\boldsymbol{\alpha}$ (see Eq. (9)).

In the latter frame, given a testing region $\mathbf{z}$ with the same size of $w \times h$, KCF uses the circular shifting on $\mathbf{z}$ again and gets the observed samples $\{\mathbf{z}_j\}_{j=1}^m$. Based on Eq.(8), a response y'_j for an observation $\mathbf{z}_j$ is computed by

$$y'_j = f\left(\varphi\left(\mathbf{z}_j\right)\right) = \left\langle \sum_{i=1}^m \alpha_i \varphi\left(\mathbf{x}_i\right), \varphi\left(\mathbf{z}_j\right) \right\rangle = \sum_{i=1}^m \alpha_i \kappa\left(\mathbf{x}_i, \mathbf{z}_j\right). \tag{15}$$

Using the Gaussian kernel as the Eq. (6), the element of the kernel matrix is computed by

$$\begin{aligned} \kappa\left(\mathbf{x}_i, \mathbf{z}_j\right) &= exp\left(-\frac{1}{\sigma_k^2}\left(\left\|\mathbf{x}_i\right\|^2 + \left\|\mathbf{z}_j\right\|^2 - 2\mathbf{x}_i^T \mathbf{z}_j\right)\right) \\ &= exp\left(-\frac{1}{\sigma_k^2}\left\|\mathbf{x}_i - \mathbf{z}_j\right\|^2\right). \end{aligned} \tag{16}$$

.

It is noticed that in Eq.(16), the exponent inside the $exp()$ function is a negative standardized Euclidean distance between $\mathbf{x}_i$ and $\mathbf{z}_j$. If $\mathbf{z}_j$ is similar to the $\mathbf{x}_i$ in the Euclidean space, $\kappa\left(\mathbf{x}_i, \mathbf{z}_j\right)$ would get great scalar value. This kind of function is called likelihood function in filtering based visual tracking [41]. Here, likelihood function is defined as a function that can be evaluated up to a scalar, given arbitrary model of observations. The strength of the scalar shows how likely the observation is made on the true target. In common visual tracker, there is only one reference sample $\mathbf{x}$. All the candidates $\{\mathbf{z}_j\}_{j=1}^m$ are compared to $\mathbf{x}$. Candidate $\mathbf{z}_j$ with the highest κ value is considered to be the feature of the target in the Maximum Likelihood criterion. If a suitable prior density is given, candidate with non-trial prior and good κ value would be the estimation according to the Maximum A Posterior (MAP) criterion [38].

By contrast, KCF uses m reference samples $\{\mathbf{x}_i\}_{i=1}^m$ to test m observations $\{\mathbf{z}_j\}_{j=1}^m$. For one observation, similarities between all the referred samples and $\mathbf{z}_j$ are measured in matrix $\boldsymbol{K}_{xz_j}[:]$ = $[\kappa(\mathbf{x}_1, \mathbf{z}_j), ..., \kappa(\mathbf{x}_m, \mathbf{z}_j)]^T$ (see Eq. (16)). Here we add $[:]$ to sequentialize the kernel matrix into the vector. Furthermore, A pre-trained model $\boldsymbol{\alpha}[:] = [\alpha_1, ..., \alpha_m]^T$ weights the similarity values κ and sums them together as y'_j in Eq. (15). y'_j would get the great value, only when the measured $\boldsymbol{K}_{xz_j}$ has the similar spatial distribution as the referred $\boldsymbol{K}_{xx}$. Finding the maximum value y'_p in $\{y'_j\}_{j=1}^m$ (which are aligned in matrix $\mathbf{Y}'$), KCF chooses the location of y'_p as the estimation.

Therefore Eq. (15) maps each observation $\mathbf{z}_j$ to a scalar y'_j. The strength of y'_j shows the probability of the observation $\mathbf{z}_j$ is extracted from the true target. **Eq. (15) can be viewed as the more advanced likelihood function. It measures two-fold similarities: one is the similarity between a candidate $\mathbf{z}_j$ and a referred sample $\mathbf{x}_i$ in the Euclidean space. This measurement is the common likelihood function in the filtering based visual tracking [41].**

The another similarity measures the difference between two kernel matrix $\boldsymbol{K}_{xz_j}$ and $\boldsymbol{K}_{xx}$. Using the location of the peak value y'_p in the $\{y'\}_{i=1}^m$, is equivalent to the Maximum Likelihood estimation. In addition, if the observation $\mathbf{z}$ is similar to $\mathbf{x}$, we could prove that the response $\mathbf{Y}'$ of KCF has similar distribution like the training label set $\mathbf{Y}$.

Since all the $\mathbf{x}_i$ and $\mathbf{z}_j$ are circular shifting in space, $\{y'_j\}_{j=1}^m$ could also be aligned to the matrix $\mathbf{Y}'$ with the same size as $\mathbf{Y}$, i.e. $w \times h$. Exploiting the circular structure, the computing of $\mathbf{Y}'$ can be accelerated by the FFT (see Eq. 11). Combining Eq. (9) and Eq. (11), we have the following Eq. (17) to compute the classification response in the frequency domain. Here, the multiply and division in the frequency domain are element-wise in this subsection.

$$\mathcal{F}\left(\mathbf{Y}'\right) = \frac{\mathcal{F}^*\left(\mathbf{K}_{xz}\right)}{\mathcal{F}^*\left(\mathbf{K}_{xx}\right) + \lambda} \mathcal{F}\left(\mathbf{Y}\right) \tag{17}$$

If we choose the simpler inner production as the kernel function, which means $\varphi(\mathbf{x}) = \mathbf{x}$, $\kappa(\mathbf{x}, \mathbf{x}) = \mathbf{x}^T\mathbf{x}$,

$$\mathbf{K}_{xx} = \mathcal{F}^{-1}\left(\mathcal{F}^*\left(\mathbf{x}\right)\mathcal{F}\left(\mathbf{x}\right)\right), \tag{18}$$

and

$$\mathbf{K}_{xz} = \mathcal{F}^{-1}\left(\mathcal{F}^*\left(\mathbf{x}\right)\mathcal{F}\left(\mathbf{z}\right)\right), \tag{19}$$

we have the response equations as follows:

$$\mathcal{F}\left(\mathbf{Y}'\right) = \frac{\mathcal{F}^*\left(\mathbf{x}\right)\mathcal{F}\left(\mathbf{z}\right)}{\mathcal{F}\left(\mathbf{x}\right)\mathcal{F}^*\left(\mathbf{x}\right) + \lambda} \mathcal{F}\left(\mathbf{Y}\right). \tag{20}$$

.

If $\mathbf{x}$ gets varied intensities, modulus of $\mathcal{F}\left(\mathbf{x}\right)$ is far bigger than the little regularized value λ. Omit the λ in Eq. (20), we get a simple but meaningful equation:

$$\mathcal{F}\left(\mathbf{Y}'\right) = \frac{\mathcal{F}\left(\mathbf{z}\right)}{\mathcal{F}\left(\mathbf{x}\right)} \mathcal{F}\left(\mathbf{Y}\right) \tag{21}$$

Given the Gaussian label $\mathbf{Y}$ with covariance $\sigma_y^2\boldsymbol{I}_2$ ($\boldsymbol{I}_2$ denotes the identity matrix with size 2×2, σ_y is a little standard deviation for the isotropic 2d Gaussian) in spatial domain [19], the $\mathcal{F}\left(\mathbf{Y}\right)$ in frequency domain holds the Gaussian form too, following the properties of Fourier transformation.

During the tracking, if the target moves without appearance varying. Each candidate $\mathbf{z}$ can be regarded as the translation of the reference $\mathbf{x}$, i.e. $\mathbf{z}(\boldsymbol{\upsilon}) = \mathbf{x}(\boldsymbol{\upsilon} + \boldsymbol{\delta})$. Here the $\boldsymbol{\upsilon} = [\gamma, \theta]^T$ is a

position variable in the *range-azimuth* coordinates, and the $\boldsymbol{\delta}$ denotes the offset vector. Put the translated form of $\mathbf{z}$ into the Eq. (21), the response matrix $\mathbf{Y}'$ in the frequency is given by:

$$\begin{aligned}\mathcal{F}\left(\mathbf{Y}'(\boldsymbol{\upsilon})\right) &= \frac{\mathcal{F}\left(\mathbf{x}(\boldsymbol{\upsilon}+\boldsymbol{\delta})\right)}{\mathcal{F}\left(\mathbf{x}\right)}\mathcal{F}\left(\mathbf{Y}(\boldsymbol{\upsilon})\right) \\ &= \frac{e^{jw\boldsymbol{\delta}}\mathcal{F}\left(\mathbf{x}\right)}{\mathcal{F}\left(\mathbf{x}\right)}\mathcal{F}\left(\mathbf{Y}(\boldsymbol{\upsilon})\right) \\ &= e^{jw\boldsymbol{\delta}}\mathcal{F}\left(\mathbf{Y}(\boldsymbol{\upsilon})\right). \\ &= \mathcal{F}\left(\mathbf{Y}\left(\boldsymbol{\upsilon}+\boldsymbol{\delta}\right)\right)\end{aligned} \tag{22}$$

Using the inverse Fourier transformation, we get $\mathbf{Y}'\left(\boldsymbol{\upsilon}\right) = \mathbf{Y}\left(\boldsymbol{\upsilon}+\boldsymbol{\delta}\right)$. It means that if the tested input $\mathbf{z}$ is the translation of the training sample, the output label $\mathbf{Y}'$ would maintain the translated form of $\mathbf{Y}$. Recall that $\mathbf{Y}$ is a needle-shaped Gaussian. So, **if z is very similar to the reference x, it brings not only the great peak value y'_p in $\mathbf{Y}'$, but also makes the $\mathbf{Y}'$ has the similar compact Gaussian distribution like the training label matrix $\mathbf{Y}$.**

B. Adding Peak to Side-lobe Ratio (PSR) to monitor the status of KCF

1) Peak to Side-lobe Ratio

If we view the KCF in the maximum likelihood perspective, the successful tracking of KCF highly depends on the likelihood distribution $\mathbf{Y}'$. A perfect estimation would get a peak value y'_p equal to 1 and a needle-shaped Gaussian distribution for $\mathbf{Y}'$. This is to say, the peak value and the concentration of the $\mathbf{Y}'$ measure the similarity between the candidate $\mathbf{z}$ and the reference $\mathbf{x}$, and also reflect the reliability of the tracking.

In this paper we use the peak to side-lobe ratio (PSR) [42] to quantitatively evaluate the reliability. Here the peak means the maximum value y'_p in $\mathbf{Y}'$. An rectangle window around the peak is selected as the peak region. The side-lobe is the rest part of $\mathbf{Y}'$ excluding the peak region. The PSR score is then computed as:

$$s = (y'_p - \mu_s)/\sigma_s. \tag{23}$$

μ_s and σ_s are the mean and standard deviation of the side-lobe respectively. The higher the PSR is, the more energy of $\mathbf{Y}'$ is concentrated in the peak region, the less μ_s and σ_s are in the side-lobe of the matrix. It indicates that the KCF works very well. Otherwise, the PSR score s

drops down, the peak value y'_p goes down and the form of $\mathbf{Y}'$ is not constrained. It indicates that the feature of the observation is varying from the reference. The changing of feature increases the distance between the candidate and the reference samples in the similarity measuring space. As we have discussed in the subsection III-D, the increasing distance between the observation and the reference will violate the prerequisites for the running average. Therefore, we use a threshold value S_{th} on PSR to measure the status of the KCF tracker. When the PSR score s is less than the S_{th}, we stop the running average both on the reference updating and online training(see Algorithm 1).

Algorithm 1: KCF with PSR score

Input:

- $\boldsymbol{b}_0$: given bounding rectangle of the target in the initial frame.
- $\mathbf{Y}$: regression label in needle-shaped 2d Gaussian
- S_{th}: threshold of the PSR score

Output:

- y'_p: the maximum value of the classification response $\mathbf{Y}'$
- s : PSR score of $\mathbf{Y}'$
- $\hat{\boldsymbol{b}}$: estimated bounding rectangle

if *in the initial frame* **then**
  Extracting the training image patch $\mathbf{x}$ in $\boldsymbol{b}_0$.
  Learning $\boldsymbol{\alpha}_0$ in the first frame using Eq. (9).
  $\hat{\boldsymbol{b}} = \boldsymbol{b}_0 = [\gamma_0, \theta_0, w_0, h_0]^T$
end
if *in the latter frame* **then**
  Extracting the testing image patch $\mathbf{z}$ in $\hat{\boldsymbol{b}}$.
  Computing the response of classification $\mathbf{Y}'$ using Eq. (11).
  Finding the position $[\gamma_p, \theta_p]$ of the maximum value y'_p in $\mathbf{Y}'$.
  Calculating the PSR score s of $\mathbf{Y}'$ by Eq. (23).
  if $s \geq S_{th}$ **then**
    Training $\boldsymbol{\alpha_t}$ in current frame using Eq. (9)
    updating $\boldsymbol{\alpha_t}$ by Eq. (14)
    updating $\mathbf{x}_t$ by Eq. (13)
  end
  $\hat{\boldsymbol{b}} = [\gamma_p, \theta_p, w_0, h_0]^T$
end
return y'_p, s *and* $\hat{\mathbf{b}}$

In Algorithm 1, in the initial frame, the simple image patch $\mathbf{x}$ in $\boldsymbol{b}_0$ is circular shifted to

generate dense features $\{\mathbf{x}_i\}$ as training samples. Making the feature in $\{\mathbf{x}_i\}$ and label in $\mathbf{Y}$ as feature-label pairs, the key parameter $\boldsymbol{\alpha}_0$ of the classifier is trained. Then in the latter frame, dense features $\{\mathbf{z}_i\}$, which are generating around the previous location are tested by the classifier function. Location of the maximum of the classification response $\mathbf{Y}'$ is viewed as the estimation $\hat{\boldsymbol{b}}$. It is worth noting that once the KCF initialize reference $\mathbf{x}$ in the initial frame, it does not change the scale (width and height) of the target. KCF only refreshes the position of the bounding rectangle $\hat{\boldsymbol{b}}$.

2) An Relay tracking example shows how PSR varies with the tracking precision

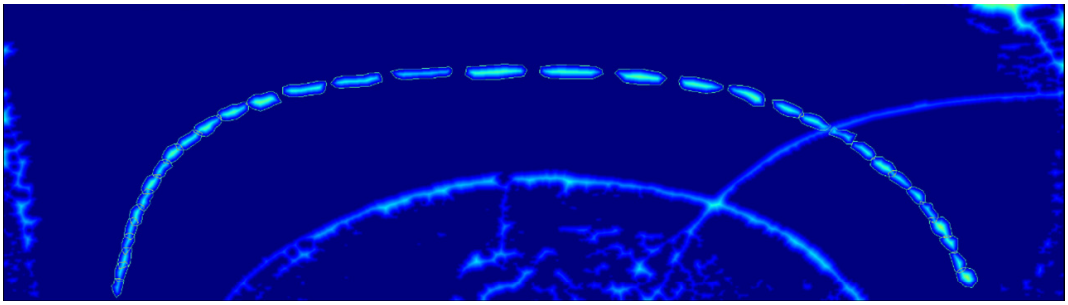
If the appearance of the target is relatively stable, the single KCF tracker would locate the target smoothly. But in real maritime surveillance, the appearance is only stable in a short term and the outer interruption is not ignorable in the long term. Here we relay tracks a ship named Alice by three independent KCF trackers to show how the PSR score of each KCF varies with the tracking precision.

Taking the ship tracking in Fig. 3 (a) for example (When the ship is approaching the top-center radar site and then leaving, the scale of the ship varies drastically), single KCF tracker named as T1, initiated at the 60th frame, would drift away in the 199th frame. However, if we manually initiate a new tracker T2 at the frame #200, T2 succeeds to track the ship until it is occluded by the bridge clutter in the 300th frame. Again, when a new KCF tracker T3 starts at the 313th frame, it follows the object to the end. In Fig. 3 (b) , we record the PSR scores and center location error (CLE) for T1 in frames [60-199], for T2 in frames [200-312], and for T3 in frames [313-400].

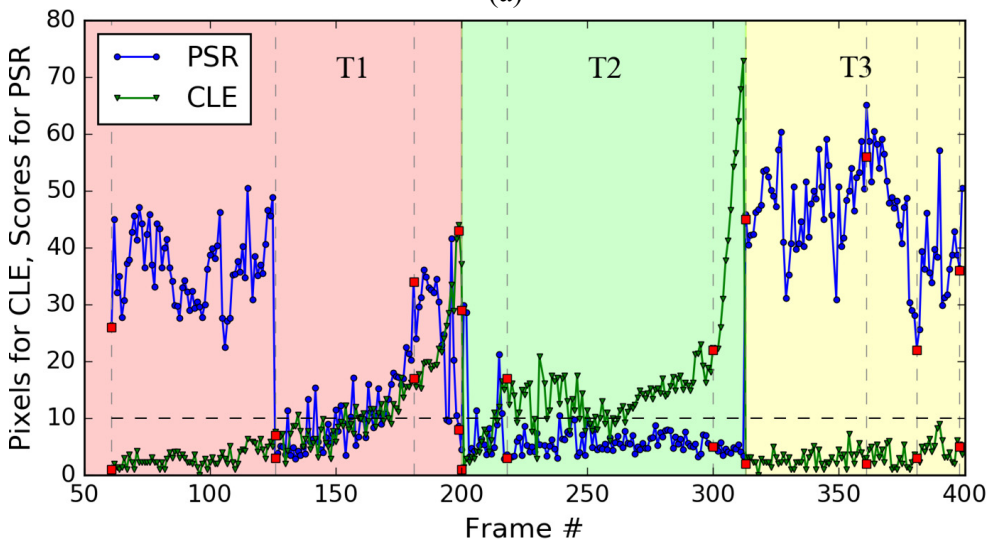
Fig. 4 shows the results of the three trackers in typical frames. It illustrates the classification response $\mathbf{Y}'$ (first row of each subfigure), estimated location (in colorful rectangle of the second row of each subfigure, white rectangle denotes the ground truth location) of T1, T2, and T3. PSR scores and corresponding CLE pixels are given in each sub figure.

It is noted that when the appearance of the ship varies or when the target is occluded by other ship, the PSR score drops down quickly and the tracker drifts away with increasing CLE. However, if we build a new tracker when the PSR of the old tracker drops down, *only 3 manually* initiated KCF trackers would be able to complete relay tracking the ship in such tough environment.

Why can not the single KCF tracker insist to the end, while 3 relayed KCF tracker could?



(a)



(b)

Fig. 3: Relay tracking results on a ship. (a) The trajectory of the ship ranges from frame #60 to #400 of the radar dataset. (b) Center location error (CLE) and PSR scores of three relay trackers. T1 tracks at the first stage from frame #60 to #199; T2 begins at #200 and ends at #312; T3 is initiated at frame #313. The red square marks highlight the PSR scores and CLE pixels for the selected frames in Figure 4. It is observable that when PSR scores below 10, the CLE would increase in the subsequent frames.

This is because, Alice get multiple models of feature. We divide it into 3 approximately. These features are captured when she is heading to, running along and leaving away the radar site. If only one KCF is using, the multiple models of the features are simply combined by the running average. Averaging spoils the discriminant of the feature in the key state. The eroded reference further effects the classifier's online training. As a consequence, the KCF could not complete the tracking in the long term. **This inspires the basic idea of this paper: KCF is fast but reliable**

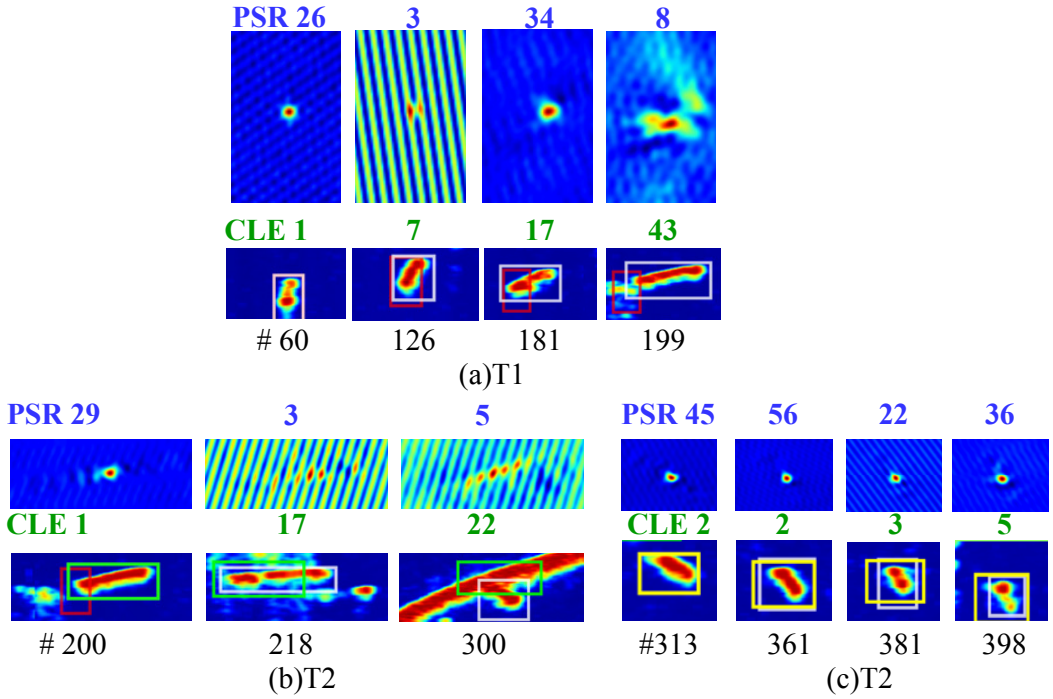


Fig. 4: Response image and tracking results for the (a) T1, (b) T2, (c) T3 in typical frames. The tracking rectangle of T1, T2, T3 is labeled in red, green, yellow respectively. The white rectangle is for the ground truth. PSR scores and pixels of CLE are titled on the top of each sub figure. Frame numbers are labeled at the bottom of each sub-images. It is noted that the compactness of response map reveals the accuracy of the KCF tracker.

in short term. If multiple KCFs are initiated in the same object at different time steps, integrating these trackers gets the chance to locate the object successfully in long term. In following subsections we describe how to fuse multiple trackers and how to *automatically* initiate new tracker.

C. Fusing the multiple KCF trackers via the Maximum Likelihood criterion and the approximation

The relay tracking in the subsection IV-B2 shows that connecting the trajectories of 3 independent KCF trackers in different time step, a ship is able to be correctly tracked in the rough environment. In the relay tracking experiment, the KCF tracker is **manually chosen at the key frame**. In each time step, only one tracker is working. It is still **vulnerable** (The average PSR

score of the $T2$ tracker is below 10). If multiple KCF trackers are working together at the same time step, the fused estimation would definitely enhance the robustness of the tracking in the long term. In this section, we fuse the multiple KCF trackers via the Maximum Likelihood approach.

1) Problem formulation

Let $\boldsymbol{v}$ ($\boldsymbol{v} \in \mathbf{V}$) be the position variable of the target at certain time step. $\mathbf{V}$ is all the positions in the search region ($w \times h$) of the tracker. There are N KCF trackers with $\{T_i\}_{i=1}^{N}$ observations. Given a location $\boldsymbol{v}$, the probability of $\boldsymbol{v}$ being the center of the target measured by each tracker is the individual likelihood $p\left(T_i|\boldsymbol{v}\right)$. Since each individual observation is independent, the joint likelihood distribution is computed by taking the product of the probability functions for each observation

$$p\left(T_1, ..., T_N|\boldsymbol{v}\right) = \prod_{i=1}^{N} p\left(T_i|\boldsymbol{v}\right). \tag{24}$$

If we consider the response of each KCF tracker as the observation from one sensor, the computing of the joint likelihood function of the multiple KCF trackers is similar to multi-sensor data fusion in [43].

In KCF the individual likelihood distribution is equal to the response matrix, i.e. $p\left(T_i|\boldsymbol{v}\right) = \mathbf{Y}_i'\left(\boldsymbol{v}\right)$. Replace the joint likelihood distribution by the fused response $\mathbf{Y}'$, Eq. (24) is rewritten by

$$\mathbf{Y}'(\boldsymbol{v}) = \prod_{i=1}^{N} \mathbf{Y}_i'(\boldsymbol{v}). \tag{25}$$

The objective of the fusing is to find the $\hat{\boldsymbol{v}}$ which maximize the $\mathbf{Y}'(\boldsymbol{v})$:

$$\hat{\boldsymbol{v}} = \arg\max_{\boldsymbol{v}} \prod_{i=1}^{N} \mathbf{Y}_i'(\boldsymbol{v}). \tag{26}$$

If we directly do the computation on Eq.(26), it will cost about $N \times w \times h$ times multiplications and the maximum finding on the $w \times h$ region. Assuming that fusing is only operated on the reliable trackers with high PSR scores, we could evaluate the maximum analytically by using the parameters of the individual likelihood distribution.

2) Approximation

As we have discussed in the subsection IV-B, if the PSR score of a KCF tracker is high, the response matrix $\mathbf{Y}_i'$ is compact and has the similar needle-shaped Gaussian as the training label

$\mathbf{Y}$. So we make the assumption that, if the s_i is high enough ($s_i \geq 10$ in the experiment), the position variable $\boldsymbol{v}$ in the likelihood space of T_i is a constrained 2d Gaussian distribution

$$\mathbf{Y}_i'(\boldsymbol{v}) \sim N(\boldsymbol{\mu_i}, \sigma_i^2 \boldsymbol{I}_2). \tag{27}$$

Here the $\boldsymbol{\mu_i}$ and $\sigma_i^2 \boldsymbol{I}_2$ ($\boldsymbol{I}_2$ denotes the identity matrix with size 2×2) are the mean and covariance respectively. For the ith tracker, the probability of $\boldsymbol{v}$ being the center position of the target has the form:

$$\mathbf{Y}_i'(\boldsymbol{v}) = \frac{1}{2\pi\sigma_i^2} exp\left(\frac{-(\boldsymbol{v}-\boldsymbol{\mu}_i)^T(\boldsymbol{v}-\boldsymbol{\mu}_i)}{2\sigma_i^2}\right). \tag{28}$$

For each KCF tracker, $\boldsymbol{v}_{p_i}$ denotes the location of the maximum value y'_{p_i} in the response matrix $\mathbf{Y}'$. So the mean $\boldsymbol{\mu_i}$ is equal to $\boldsymbol{v}_{p_i}$, and the σ_i is inversely proportional to the y'_{p_i},

$$y'_{p_i} = 1/(2\pi\sigma_i^2). \tag{29}$$

Therefore, given N independent Gaussian shaped response $\{\mathbf{Y}_i'(\boldsymbol{v})\}_{i=1}^N$, the objective of the fusing of the multiple KCF trackers is to find the $\hat{\boldsymbol{v}}$ which maximizes the overall likelihood function:

$$\mathbf{Y}'(\boldsymbol{v}) = \prod_{i=1}^{N} \mathbf{Y}_i'(\boldsymbol{v}) = \prod_{i=1}^{N} \frac{1}{2\pi\sigma_i^2} exp\left(\frac{-(\boldsymbol{v}-\boldsymbol{\mu}_i)^T(\boldsymbol{v}-\boldsymbol{\mu}_i)}{2\sigma_i^2}\right). \tag{30}$$

The log likelihood function is given by:

$$\ln(\mathbf{Y}'(\boldsymbol{v})) = \ln(C) - \sum_{i=1}^{N} \frac{(\boldsymbol{v}-\boldsymbol{\mu}_i)^T(\boldsymbol{v}-\boldsymbol{\mu}_i)}{2\sigma_i^2}. \tag{31}$$

Here the C is used to define the items that are irrelevant to the position variable $\boldsymbol{v}$.

The derivative of the log likelihood with respect to $\boldsymbol{v}$ is given by:

$$\frac{\partial}{\partial \boldsymbol{v}} \ln(\mathbf{Y}'(\boldsymbol{v})) = \sum_{i=1}^{N} \frac{(\boldsymbol{v}-\boldsymbol{\mu}_i)}{\sigma_i^2}. \tag{32}$$

Setting the derivative to zero, we obtain the solution for the maximum likelihood estimation of the position given by

$$\hat{\boldsymbol{v}} = \frac{\sum_{i=1}^{N} (\boldsymbol{\mu}_i/\sigma_i^2)}{\sum_{i=1}^{N} 1/\sigma_i^2}. \tag{33}$$

Since the $\boldsymbol{\mu}_i = \boldsymbol{v}_{p_i}$ and put the Eq. (29) into the Eq. (33), the solution is also given by

$$\hat{\boldsymbol{v}} = \frac{\sum_{i=1}^{N} \left(\boldsymbol{v}_{p_i} y'_{p_i}\right)}{\sum_{i=1}^{N} y'_{p_i}}. \tag{34}$$

For fusing width w_i and height h_i of the ith tracker, we reuse the same weights in Eq. (34) . Therefore, given the estimation $\tilde{\boldsymbol{b}}_i = [\gamma_i, \theta_i, w_i, h_i]^T$ of a KCF component, result of the fusing is computed by

$$\begin{aligned}\hat{\boldsymbol{b}} &= \frac{\sum_{i=1}^{N}\left(\tilde{\boldsymbol{b}}_i y'_{p_i}\right)}{\sum_{i=1}^{N} y'_{p_i}} \\ &= \sum_{i=1}^{N} w_i \tilde{\boldsymbol{b}}_i.\end{aligned} \tag{35}$$

Here the

$$w_i = \frac{y'_{p_i}}{\sum_{i=1}^{N} y'_{p_i}}.$$

This estimation of $\hat{\boldsymbol{b}}$ saves a lot computing time compared to the element wise multiplication in Eq. (25). The average searching region is (100×200) for the matrix $\mathbf{Y}'_i$ and average tracker number is $N = 3$ in our experiment. Since the $\tilde{\boldsymbol{b}}_i$ and y'_{p_i} are directly the output of each KCF tracker, the computing of fusing in Eq. (35) only needs 3 multiplications. By contrast, Eq.(26) needs about 60000 element wise multiplication and an extra maximum searching.

For one KCF tracker, the complexity is decided by the FFT operations and is $O(M^2 \log M)$ for a patch with the size $M \times M$ (M denotes the maximum width or height of the patch, see [19]). So the computing complexity of MKCF is about $NM^2 \log M$ (N is the number of the fused trackers).

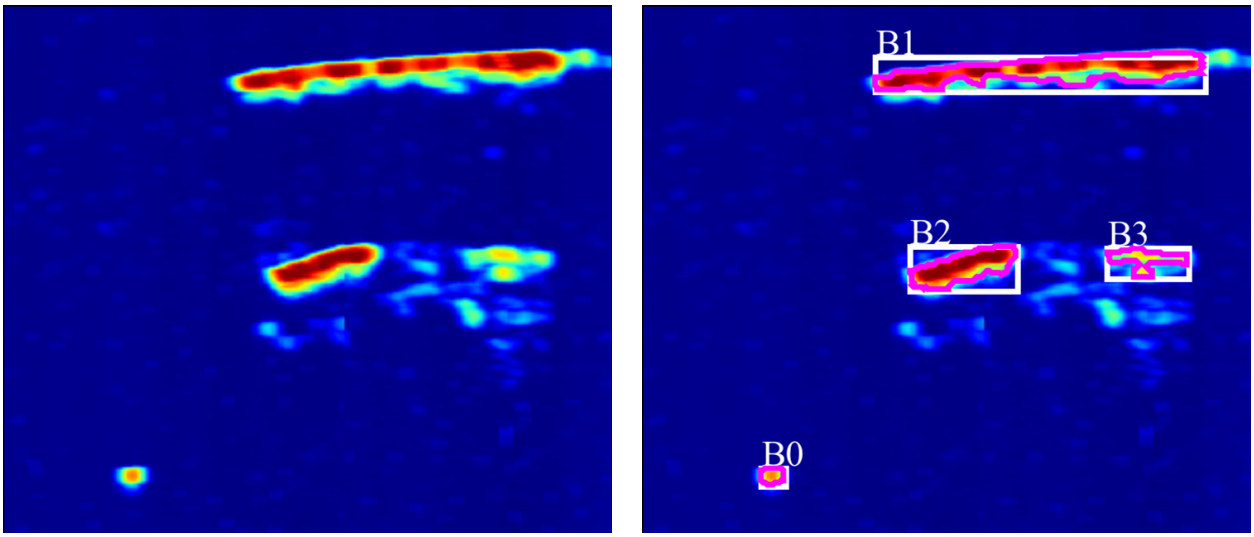
D. Object segmentation for automatic initializing new tracker

In order to operate redundant trackers on the target, there is a need for automatic initializing the new KCF tracker. That means we need to do object segmentation and obtain the bounding box of the target, when an initializing is required. In this section, we describes our strategy of object segmentation for the ships in the radar image.

Marine radar captures the straight-reflected echo signal in discrete azimuth units. Accumulating the azimuth-aligned intensities in a scan, a 2D polar radar image is formatted. Details about the accumulating and formatting could be referred to [44].

Ships in inner-river are close to the radar station. They strongly reflects the radar's radio waves, which brings high intensities on the polar image (see Figure 5) . Here we don't use the Constant False Alarm Rate (CFAR) based detection method [45]. CFAR is an auto-threshold

selection method on a single direction. If the object is consists of echoes across multiple azimuth units, further merging step is needed for tracking. As we concern the region, the object is directly segmented by marker-based watershed algorithm [46]. Details of the implementation are referred to the manual [47].



(a) Source (b) Blobs

Fig. 5: Segmentation on the radar image. (a) source radar image, (b) four segmented blobs. Bounding box of the blob is labeled in white rectangle. Contour of the blob is marked in Magenta.

However this segmentation is simple but not reliable. Segmenting the target in the clutter would get a list of blobs. Which blob contains the true target? More clues are needed to do the associating. Given the estimated bounding box set $\{\tilde{\boldsymbol{b}}_i\}_{i=1}^{N}$ of the multiple trackers, intuitively the blob, which is overlapped with most of the bounding boxes, is the most like to be the region of the target.

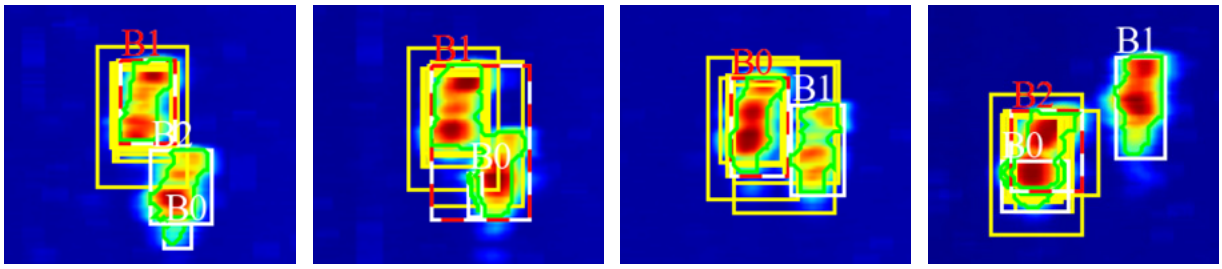
In each frame, segmenting the neighborhood of the previous box $\hat{\boldsymbol{b}}^{-}$, we get a list of blobs. Based on the intersection-over-union (IOU) between the segmented polygon and the bounding boxes of the selected trackers, we get a IOU matrix table. Given two shapes A and B, IOU is the intersected area divided by their union area: IOU = Area(A $\cap$ B)/Area(A $\cup$ B) .

Each blob computes the IOU with all the bounding boxes of the qualified trackers whose PSR score s is no less than the S_{th}. Assuming that there is 3 qualified trackers and 7 blobs, We list all the IOU values between the tracking boxes and segmented blobs in Table II, where

467 T represents tracker, B denotes blob. If the maximum average IOU 0.53 of $B2$ is no less than
468 the IOU threshold O_{th}, the bounding rectangle $\boldsymbol{r}$ of $B2$ is used to initiate a new tracker. Please
469 note that only those trackers with high PSR score ($s \geq S_{th}$) get the right to vote the blob.

TABLE II: An Example of IOU matrix for 7 blobs and 3 trackers. The maximum value of each row is marked in red color.

	B1	B2	B3	B4	B5	B6	B7
T1	0.5	0.4	0.00	0.00	0.00	0.00	0.00
T2	0.2	0.5	0.3	0.00	0.4	0.2	0.1
T3	0.1	0.7	0.1	0.00	0.00	0.00	0.00
Average of IOU	0.27	0.53	0.13	0	0.13	0.07	0.03



(a) Overataking (b) Merging (c) Confusing (d) Splitting

Fig. 6: Segmentation and voted rectangle during the overtaking of two ships. Segmented blob is bounding with white rectangle. Voted rectangle of the blob is marked in dash red. The yellow rectangles are the locations of the qualified trackers. (a) A similar ship (below) begins overtaking the tracking target (up). Three blobs named B0, B1, B2 are evaluated by the yellow bounding boxes of the multiple trackers. B1 blob marking in dash red is chosen for initiating a new tracker. (b) The merging of two ships makes a confusing segmentation. The voted rectangle in dash red contains the two ships together. (c) The wrong and bigger rectangle initiates a confusing tracker. The rectangle now is changing from the dash red to the solid yellow. (d) When the two ships are splitting, the majority of bounding boxes (yellow) on the true target vote a refreshed blob B2 (dash red) to initialize a correct tracker. The former confusing tracker with the least PSR score is cleared. It is observed that all the qualified trackers do not contain the overtaking ship (on the right side).

470 Blob segmentation is easy to be affected by clutter interruption or ship overlapping (see Figure
471 6). When the two ships are merging, connected region causes the wrong segmentation in a bigger

rectangle and initiates a confusing tracker. Once the two ships are splitting, newly segmented rectangle with smaller size contains the most IOU of the multiple trackers. It gives birth to a new tracker with higher PSR score, which will take the place of the confusing tracker. In fact, the tracker with high PSR score has a strong confidence of that the true location of the target is near to the tracker's estimation. Blob with the most trackers' votes has the biggest chances of containing the true target. Combining multiple trackers' confidence, the accumulating errors caused by the wrong segmentation drops.

E. Framework of the proposed MKCF

The proposed MKCF is summarized in Algorithm 2. Known $\boldsymbol{b}_0$, in the initial frame, a KCF tracker is initialized and stored in the *TrackerList*. Then in the latter frame, each tracker in the *TrackerList* completes tracking independently. All the estimated locations and the peak values of the responses are kept in the *BoxScoreList*. Later in the fusing model, $\hat{\boldsymbol{b}}$ is the weighted sum of the estimation set $\{\boldsymbol{b}_i\}$. Their weights are jointly affected by their peak values of the responses. If the *TrackerList* exceeds the limitation N_{th}, the unqualified tracker with least PSR score is deleted. It is worth noting that the allowed number of trackers N_{th} is dynamically changing in the range of $[N_{min}, N_{max}]$. When the average PSR in the tracker list is greater than the S_{th}, it means that the status of the MKCF is stable and it may need less trackers. So the N_{th} decreases 1. Otherwise, when the average PSR in the tracker list is lower than the S_{th}, it denotes that the MKCF could not be qualified to track and it needs more tracker. So the N_{th} increases 1.

Finally, segmentation and voting model would initiate a new tracker based on the bounding rectangle $\boldsymbol{r}$ of the voted blob, if the averaged IOU between the selected blob and all the trackers' boxes is bigger than O_{th}. The new tracker is added to the *TrackerList* and ready for tracking in the next frame. Since the KCF tracker is less sensitive to the scale changing, we use the voted rectangle $\boldsymbol{r}$ to replace the fusing $\hat{\boldsymbol{b}}$, if they are highly overlapped.

V. Experiment

A. Specifications of the marine radar

Experimental data was collected on August 8th, 2013 by an X-band marine radar, which is located on the bank of Yangtze River, Yichang, China (see the radar's field of view in Fig. 7).

Algorithm 2: MKCF

Input: $\boldsymbol{b}_0$, σ_y, S_{th}, O_{th}, N_{th}, N_{max}, N_{min}, *frame*.

Output: $\hat{\boldsymbol{b}}$

if *in the initial frame* **then**
  Generating needle-shaped 2d Gaussian label matrix $\mathbf{Y} \sim \mathcal{N}\left([\gamma_0, \theta_0]^T, diag\left(\sigma_y^2, \sigma_y^2\right)\right)$
  T_1 = *KCF_Init*($\boldsymbol{b}_0, \mathbf{Y}, S_{th}$,*frame*)
  TrackerList.append(T_1)
  $\hat{\boldsymbol{b}} = \boldsymbol{b}_0$
  $\hat{\boldsymbol{b}}^- = \hat{\boldsymbol{b}}$
end
if *in the latter frame* **then**
  BoxScoreList.clear()
  for i = *1 : length(TrackerList)* **do**
    T_i = *TrackerList*[i]
    $[y'_{p_i}, s_i, \boldsymbol{b}_i]$ = T_i.*tracking*(S_{th},*frame*)
    BoxScoreList.append($[y'_{p_i}, s_i, \boldsymbol{b}_i]$)
  end
  min_psr = minimum PSR in *TrackerList*.
  ave_psr = average PSR in *TrackerList*.
  if *length(TrackerList)* $\geq N_{th}$ *and* $min\_psr < S_{th}$ **then**
    TrackerList.delete(*Tracker* with the minimum PSR score)
  end
  if $ave\_psr \leq S_{th}$ **then**
    $N_{th} = min(N_{th} + 1, N_{max})$
  else
    $N_{th} = max(N_{th} - 1, N_{min})$
  end
  $\hat{\boldsymbol{b}}$ = *fuse*(*BoxScoreList*)
  BlobList = *segmentation*($\hat{\boldsymbol{b}}^-$, *frame*)
  $[\boldsymbol{r}, ave\_iou]$ = *voting*(*BlobList*, *BoxScoreList*, S_{th})
  if $ave\_iou \geq O_{th}$ **then**
    T = *KCF_Init*($\boldsymbol{r}, \mathbf{Y}, S_{th}$,*frame*)
    TrackerList.append(T)
  end
  if *is_overlapped*($\boldsymbol{r}, \hat{\mathbf{b}}$)==*TRUE* **then**
    $\hat{\boldsymbol{b}} = \boldsymbol{r}$
  end
  $\hat{\boldsymbol{b}}^- = \hat{\boldsymbol{b}}$
end

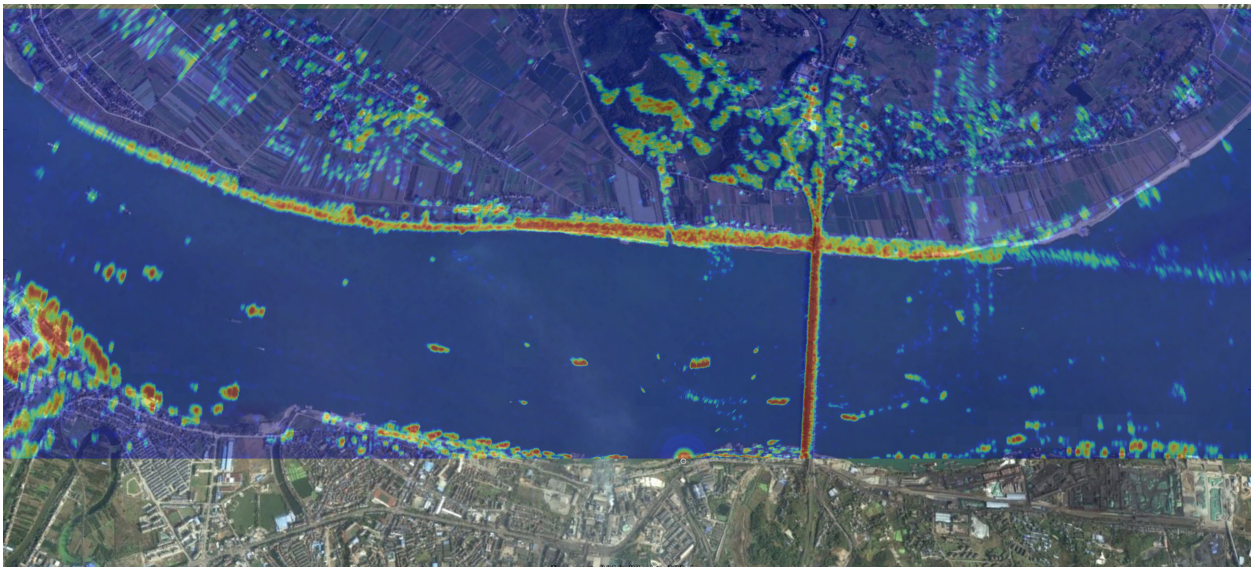


Fig. 7: Radar image overlaid on an optical image. Radar is mounted on the bank of Yangtze River, Yichang, China. (Optical image Source: $30°17'44''$ N, $111°32'05''$ E, Google Earth. Accessed on Dec. 8, 2016.)

TABLE III: X-Band Marine Radar Specifications

Antenna Length:	8ft
Beam width:	$0.95°$
Polarization:	Horizontal
Antenna rotation speed:	24 rotations per minute
Frequency:	9.4 GHz
Pulse repetition frequency (PRF):	2200 Hz
Sampling Frequency:	40 MHz
Detection range:	7680 m
Range resolution:	3.75 m
Radar cells in range:	2048
Radar cells in azimuth:	4096

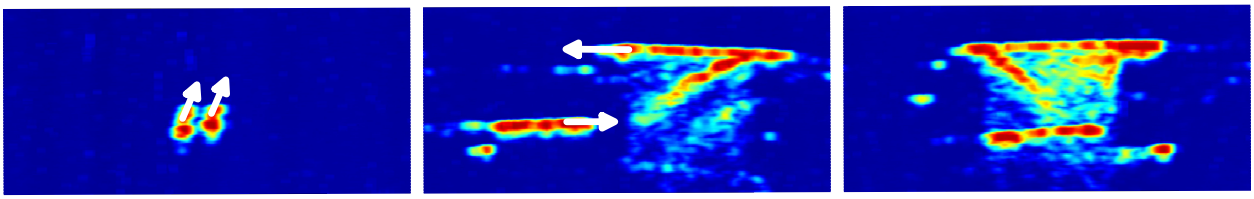
The radar has an antenna mounted on a rotor with fitted rotating speed. The bearing accuracy is about $0.95°$. The digital acquisition board works at the sampling frequency of 40 MHz. The range of surveillance is limited to 7.6 km with a high resolution (3.75m per range unit) . When the antenna scans a cycle, a frame of radar image is formatted with 2048 units in range and 4096 units in azimuth. The detailed specifications are shown in Table III.

B. Dataset overview and key parameter settings

TABLE IV: Overview of five targets in the tested data

target name	overtaking frames	head-on meeting frames	crossing bridge frames	interruption frames	min. size	max. size
Alice	78-108	213-223	290-300	192-200 215-222	23×35	26×104
Billy	213-226	102-113	143-153	-	29×20	22×132
Camen	-	189-197 212-223 247-259	-	190-196 215-222	39×35	45×170
Dolphin	81-117	167-178 188-197	235-243	190-195	29×27	27×93
Ellen	-	246-258	332-342	-	49×24	45×152

Collected data consists of 460 consecutive frames in 20 minutes. Selected five targets are named as Alice, Billy, Camen, Dolphin and Ellen respectively. Table IV gives the overview of the five targets in the tested radar data. Challenging frames in different situations are listed in the table. Maximum and minimum size (in range and azimuth units) of the targets are recorded also. Three typical cases are vividly shown in the Figure 8. We manually mark the contour of the targets in all the frames. All the vertex, center, bounding box of the polygons of the targets are computed and stored in the ground truth files.



(a) (b) (c)

Fig. 8: Three challenging situations in tracking the ships. (a) overtaking, (b) head-on meeting, (c) interruption clutter. The white arrow marks the direction of the movement.

In Tab.V we list the key parameters of the proposed MKCF. Variable w and h are the width and height of the bounding box of the target respectively.

C. Evaluation methodology

In the experiments, three evaluation criteria: mean of center location error (CLE), distance precision (DP), and overlap precision (OP) are used. In radar tracking, CLE is often used to evaluate the trackers. While in the visual tracking, DP and OP are more popular.

The first metric, mean CLE, is the average Euclidean distance between the ground-truth and the estimated center location of the target. **The smaller value of CLE means the more accurate result**.

The second metric, DP, is the percentage of frames that the CLE is smaller than a certain threshold. **More accurate tracker has higher DP value at the lower CLE threshold**.

To measure the overlap precision, we use intersection-over-union (IOU) between the bounding box of the tracker and the rectangle of the ground truth. The larger value of IOU means the more accurate result. Therefore, the third metric, OP, is now defined as the percentage of frames where the IOU surpass a range of threshold from 0 to 1. **More accurate tracker has higher OP value at the higher IOU threshold**.

D. Performance

For performance assessment, seven trackers including one traditional kalman filter, two EOT based trackers, three KCF based visual trackers and the proposed MKCF, are evaluated. Their names are given as follows:

- KF represents the **K**alman **F**ilter based object tracker.
- EOTRM is the **E**xtended **O**bject **T**racking with **R**andom **M**atrix [3].
- EOTGGIW is the **E**xtended **O**bject **T**racking with **G**amma **G**aussian **I**nverse **W**ishart density [48].

TABLE V: Parameter settings

Name	Value	Specification
σ_y	$\sqrt{w \times h}/16$	Standard deviation of the $\mathbf{Y}$ label.
σ_k	0.2	gaussian kernel bandwidth.
S_{th}	10	The minimum PSR scores for the qualified tracker.
O_{th}	0.15	The minimum IOU for generating a new tracker.
N_{th}	[3, 4, 5]	The maximum number of the fused trackers is dynamically changing.

- KCF denotes **K**ernelized **C**orrelation **F**ilter based visual tracker [19].
- SAKCF means **S**cale **A**daptive **K**ernelized **C**orrelation **F**ilter based visual tracker [21].
- MKL is the **M**ulti-**K**ernel **L**earning based KCF [23]. Since MKL works better if the input image has more color channels, we convert the intensity of radar echos (which gets only the gray channel) into the colorful heat map (which has Red, Green and Blue three color channels) with Jet color mapping. Therefore, MKL uses RGB colors and gradients as features.
- MKCF is our proposed tracker by fusion of **M**ultiple **K**ernelized **C**orrelation **F**ilters. The feature used in MKCF is only the intensity of the echos which is the same as the compared KCF.

It is worth noting that KF and EOTRM use the same segmentation method proposed in the subsection IV-D. The segmented blob which is closer to the last position is chosen as the current measurement for the target. KF only takes the centroid of the blob as input, while the EOTRM views all the points inside the blob as measurement.

TABLE VI: Consuming time per frame(**millisecond**)

	Alice	Billy	Camen	Dolphin	Ellen	Average
KF	3.2	3.0	3.3	3.3	3.0	3.2
EOTRM [3]	8.6	9.0	9.2	9.2	8.5	8.9
EOTGGIW [48]	10.6	12.3	13.6	10.8	10.0	11.4
KCF [19]	7.9	7.8	10.3	7.3	8.0	8.2
ASKCF [21]	11.4	16.0	18.2	11.8	13.2	14.1
MKL [23]	7.2	6.7	8.2	6.0	6.3	6.9
MKCF	22.1	19.5	87.8	17.2	57.9	40.9

1) Time consuming

All the trackers except MKL are implemented in Python (source code can be download at the personal Github website [1] soon after the publication, now the experimental results could be viewed online). We reuse the Matlab code of MKL [23] from the author's website. Since the author of MKL has already exploited C-compiled code for fast computing, we estimate that the time consuming in the pure Python version of MKL would be higher than the optimized Matlab.

[1]https://github.com/joeyee/MKCF

We perform the experiments on a workstation with the 8th generation Intel i7 six-core 2.60 GHz CPU and 16 GB RAM. Consuming time per frame for all the tested trackers are listed in the Table VI. For the proposed MKCF, target with bigger size or fusion with more trackers costs more computing time. The proposed MKCF trackers costs 40.9 milliseconds for single object tracking per frame in average, while the radar gets a frame in every 2500 milliseconds. The proposed MKCF has the potential to track around 50 targets simultaneously in real time without further optimization.

2) Memory usage of MKCF

We have stored the component tracker of MKCF in a tracker list. The main memory consuming are used for storing 5 variables of the KCF tracker: the reference $\mathbf{x}$, the observation $\mathbf{z}$, the label matrix $\mathbf{Y}$, the parameter matrix $\boldsymbol{\alpha}$ and the response matrix $\mathbf{Y}'$. All of them are 2D matrices with same size $w \times h$. In our experiment, the biggest searching region is defined as the size of 200×100 with 4 bytes float for each pixel. Therefore, the maximum memory for MKCF with $N_{max} = 5$ trackers costs $5 \times 5 \times 200 \times 100 \times 4$ bytes (around 1.9 Mb). It is affordable for the server with 16Gb memory.

3) Center location error

TABLE VII: Mean of center location error (CLE)(pixels)

	Alice	Billy	Camen	Dolphin	Ellen	Average
KF	691	522	**7**	649	**5**	375
EOTRM [3]	689	521	**6**	648	**5**	374
EOTGGIW [48]	689	521	**6**	648	**5**	374
KCF [19]	595	610	31	563	464	453
ASKCF [21]	579	**330**	1671	1060	808	890
MKL [23]	**274**	511	18	**362**	325	**298**
MKCF	**4**	**7**	19	**5**	**6**	**8**

Table VII lists the mean of center location error for all the trackers. In each column, the **green color** represents the first-ranked tracker with **least** mean of CLE and the **blue color** denotes the second. Since the target Camen and Ellen have not been interrupted by other clutter (there is no over-taking frame, see Table IV), precise segmentation help the KF and EOTRM to achieve comparable results. The results prove that the proposed segmentation method is effective for object detection when no interruption happens in the tested dataset.

4) Precision

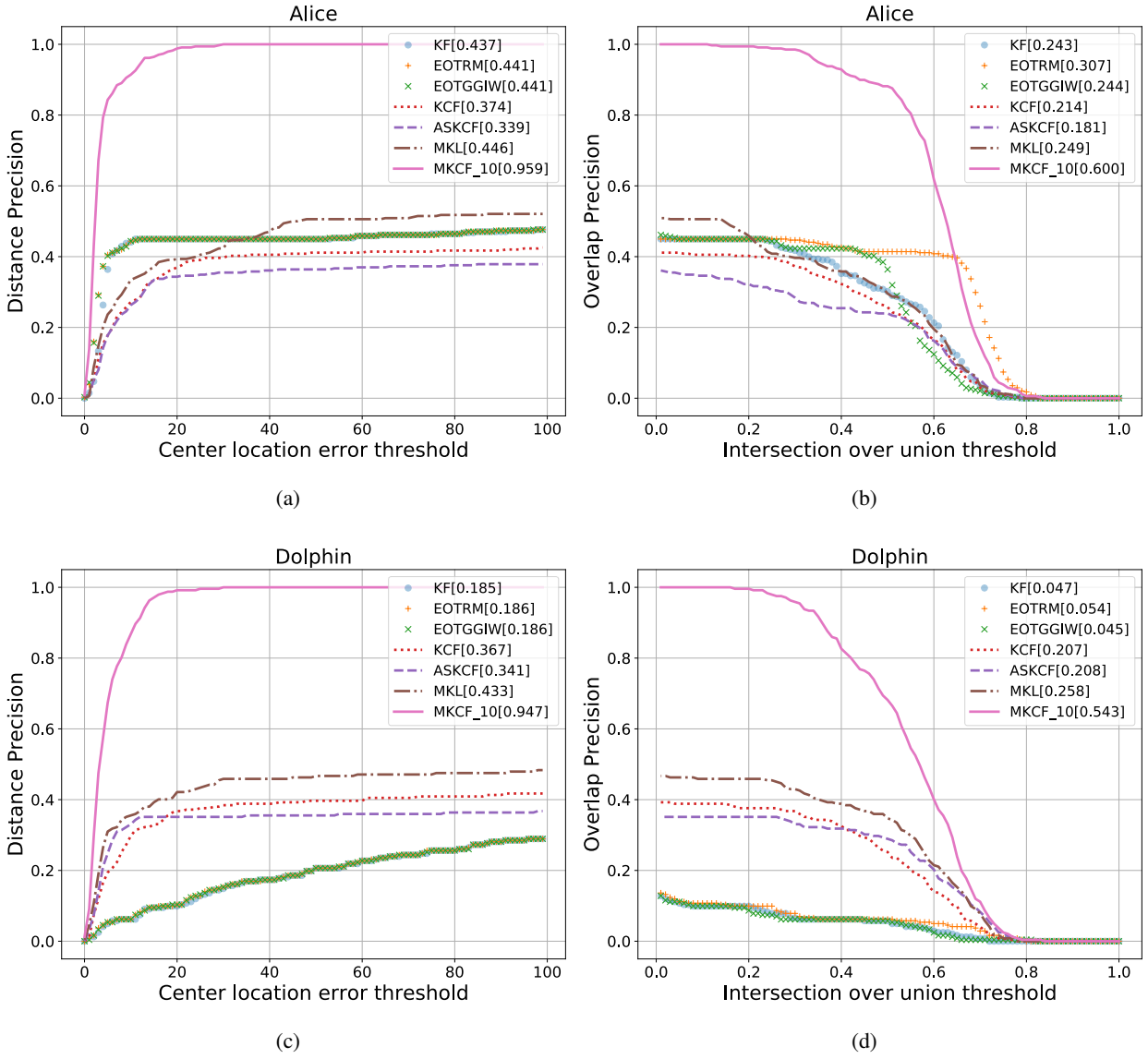


Fig. 9: Distance and overlap precision for Alice and Dolphin

Precision plot measures the percentage of successful tracking frames, given a series of thresholds (see Figure 9,10,11) . For distance precision (DP), varying center location errors are taken as the thresholds. For overlap precision (OP), different IOU values are considered. The biggest area under the curve (AUC) in the precision plot reflects the best tracker with the overall performance.

When tracking the Alice and Dolphin, only the proposed MKCF succeeds to locate the targets in all the frames. Apparently, in Figure 9, the MKCF has the biggest area under the curve. In

the legend, we record the average percentage over all the threshold. MKCF gets the best scores.

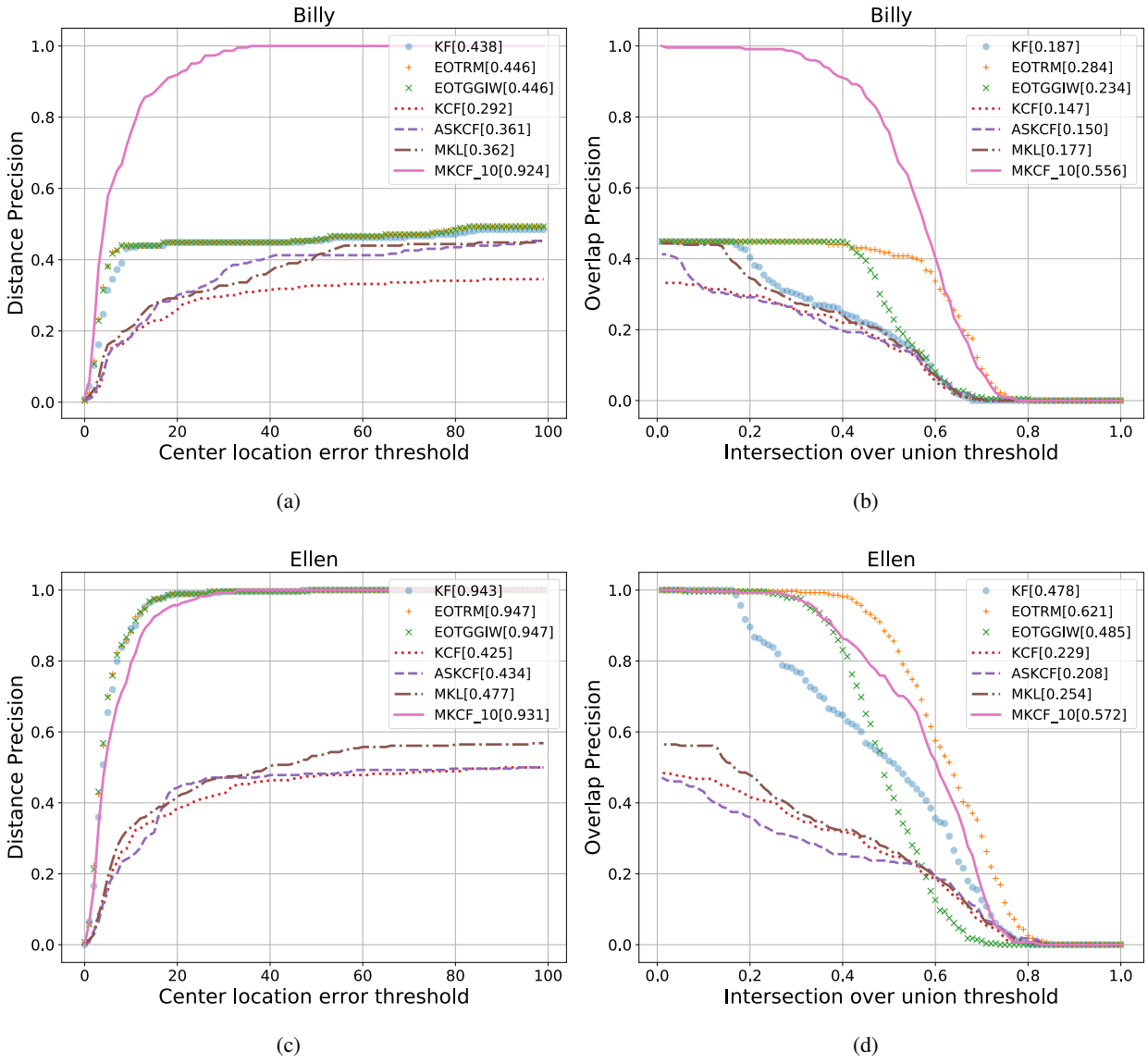
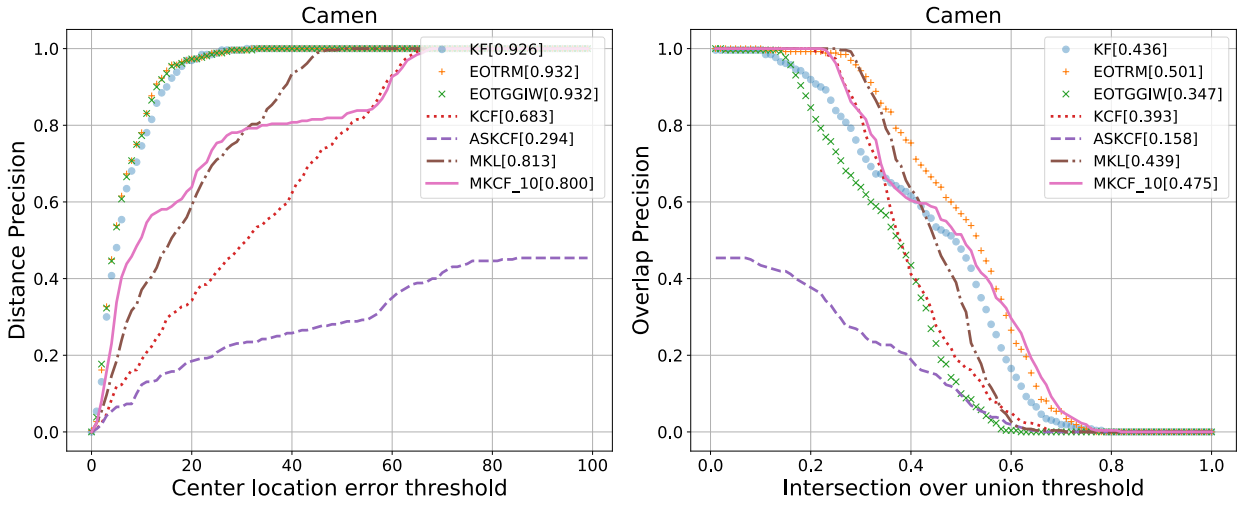


Fig. 10: Distance and overlap precision for Billy and Ellen

When tracking the Ellen (see Figure 10), EOTRM achieves comparable performance with the proposed MKCF. As mentioned before, without the interruption of clutter, blob segmentation gives EOTRM accurate measurements for estimating the center and shape of the target. Precision plot measures the overall performance. The wrong association of EOTRM in a small amount of time is not observable in the precision plot. MKCF estimates the center and shape by averaging

multiple instances. It changes the scale slower but maintains the robustness. Since KF and KCF use the fixed scale, the overlap precision degrades more sharply compared to the EOTRM and MKCF.



(a) (b)

Fig. 11: Distance and overlap precision for Camen

Table VIII and table IX list the mean of DP and OP for all the trackers. It noticed that MKCF does not rank in the top place on tracking Camen in the Table VIII (see the Fig. 11 for detail). The reason is that, Camen moves closer to the radar and has a strong multi-path clutter. KF does not consider the size, it is not affected. EOTRM and EOTGGIW model the echo cluster in Gaussian distribution which convert the irregular clutter shape into the ellipse shape. This curve fitting helps to locate the target on the real centroid. Although MKCF is not so precise on locating the centroid of Camen, the changing of size is captured by MKCF. It stills has the second overlapping score on Camen, see Table IX. In both tables, it is observed that MKCF ranks first in average.

VI. Conclusion

In this paper we propose a visual tracker framework to conduct single object tracking in a short-range marine radar scene. Integrating the appearance features of extended object, our approach improves the robustness of ship tracking in the complex environment. By fusion of

TABLE VIII: Mean of distance precision (DP$\in [0,1]$)

	Alice	Billy	Camen	Dolphi	Ellen	Average
KF	0.437	0.438	**0.926**	0.185	**0.943**	0.586
EOTRM [3]	0.441	**0.446**	**0.932**	0.186	**0.947**	**0.591**
EOTGGIW [48]	0.441	**0.446**	**0.932**	0.186	**0.947**	0.590
KCF [19]	0.374	0.292	0.683	0.367	0.425	0.428
ASKCF [21]	0.339	0.361	0.294	0.341	0.434	0.354
MKL [23]	**0.446**	0.362	0.813	**0.433**	0.477	0.506
MKCF	**0.959**	**0.924**	0.800	**0.947**	0.931	**0.912**

TABLE IX: Mean of overlap precision (OP$\in [0,1]$)

	Alice	Billy	Camen	Dolphin	Ellen	Average
KF	0.243	0.187	0.436	0.047	0.478	0.278
EOTRM [3]	**0.307**	**0.284**	**0.501**	0.054	**0.621**	**0.353**
EOTGGIW [48]	0.244	0.234	0.347	0.045	0.485	0.271
KCF [19]	0.214	0.147	0.393	0.207	0.229	0.238
ASKCF [21]	0.181	0.150	0.158	**0.208**	0.208	0.181
MKL [23]	0.249	0.177	0.439	**0.258**	0.254	0.275
MKCF	**0.600**	**0.556**	**0.475**	**0.543**	**0.572**	**0.549**

multiple correlation filters, redundant feature instances and scales enhance the reliability of tracking and segmenting. Compared to point or shape based tracking filter in the experiment, MKCF shows the advantage of target association when the target is interrupted by the clutter or other object. Also, it conquers the difficulty of scale changing, which puzzles the original KCF. Our experiments indicate that feature-based visual tracker is promised in marine radar.

Since the proposed MKCF costs more time than the conventional radar trackers, for further implementation on multiple targets tracking, optimization method of appearance learning and classifying should be considered.

Appendix A

Flow diagrams of the MKCF

In this section we illustrate the MKCF in two flow diagrams (see Fig. 12 and Fig. 13) and summarize it in the Algorithm 2. Figure 12 is for sequential overview and the Figure 13 unfolds

the detail in a single time step.

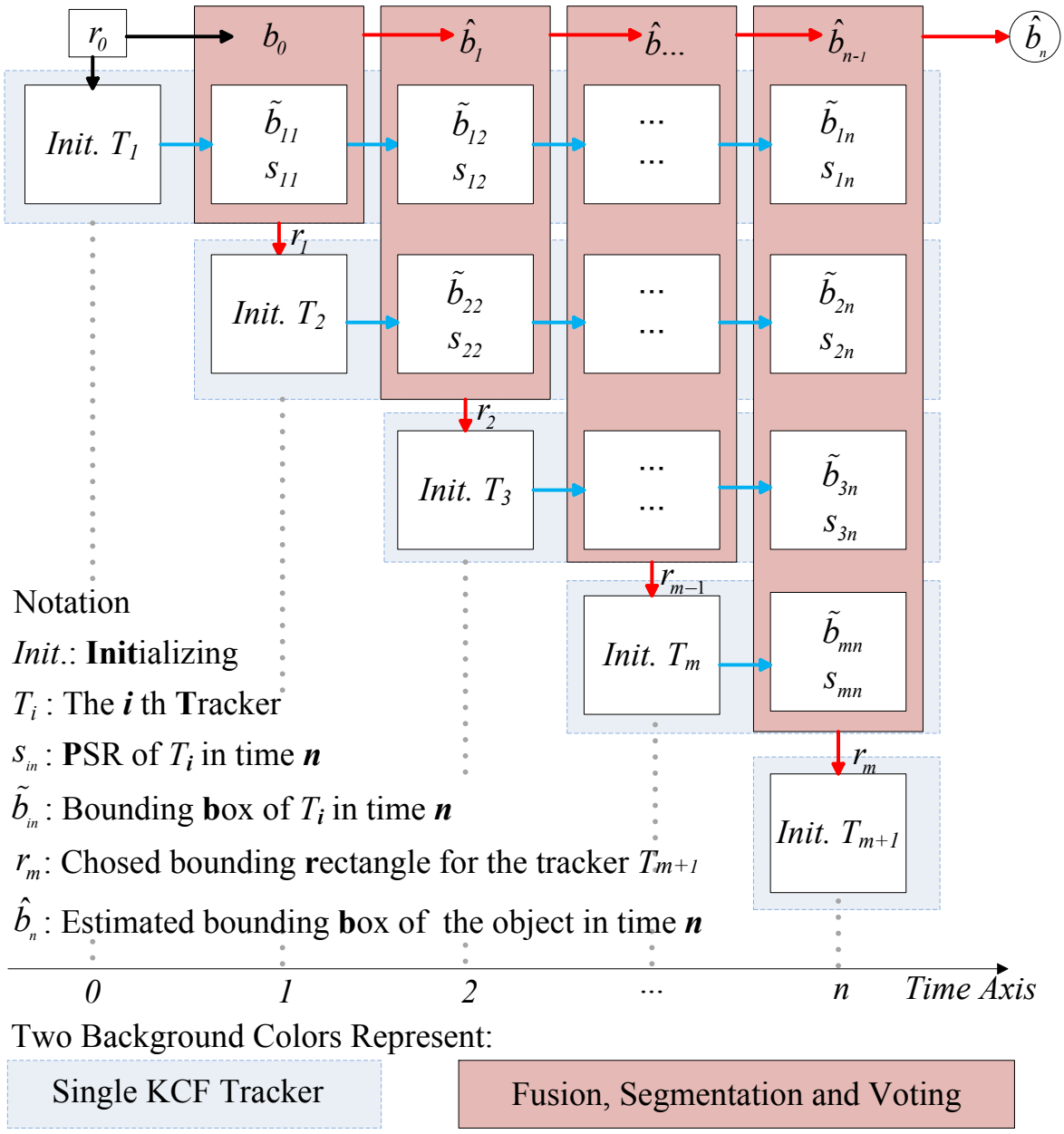


Fig. 12: The working flow of MKCF from the beginning to the time step n. In each time step, a new KCF tracker T may be initialized by the voted rectangle r. Output of the MKCF in time n is the estimated location $\hat{\boldsymbol{b}}_n$.

Figure 12 shows the working flow of MKCF from the time step 0 to n with two major components. The horizontal rectangle with light-gray background color represents a single independent KCF tracker (see Alg.1). The individual tracker T_i is initiated by given rectangle r_i in different

time step. Output of the T_i is the inferred bounding box $\tilde{\boldsymbol{b}}_{in}$, PSR scores s_{in} and peak response $y'_{p_{in}}$.

The vertical deep-brown rectangle includes the fusion (Section IV-C), segmentation and voting model (Section IV-D). In time step n, this model has three inputs: last location $\hat{\boldsymbol{b}}_{n-1}$, trackers' locations $\{\tilde{\boldsymbol{b}}_{in}\}_{i=1}^{m}$ and peak response $\{y'_{p_{in}}\}_{i=1}^{m}$. Output $\hat{\boldsymbol{b}}_n$ is inferred by fusion model. While bounding rectangle $\boldsymbol{r}_n$ of the target is the directly outcome of image segmentation and voting.

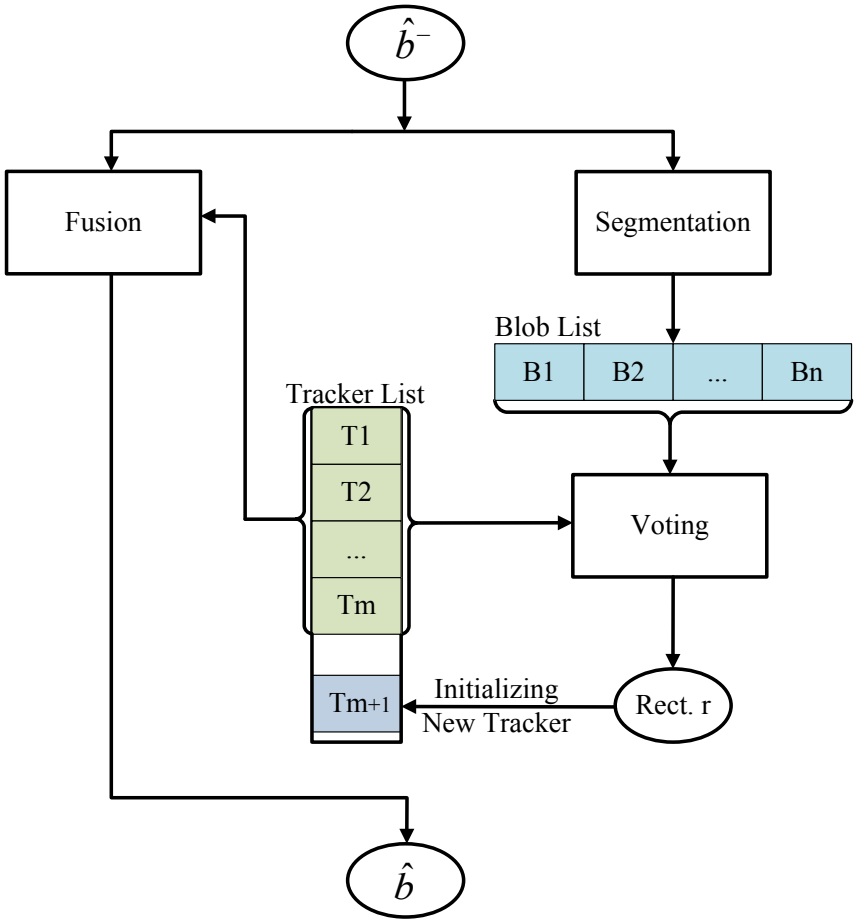


Fig. 13: The working flow of MKCF in one time stamp, consists of 3 major steps: fusion, segmentation and voting. The input $\hat{\boldsymbol{b}}^-$ is the inferred position in previous time step. Output $\hat{\boldsymbol{b}}$ is the estimated position in current time.

Figure 13 illustrates the working flow of MKCF in a certain time step. At that stage, the MKCF already gets m trackers $\{T_i\}_{i=1}^{m}$. Firstly, inputting the last location $\hat{\boldsymbol{b}}^-$, fusion model uses m trackers' locations and peak values of the responses to estimate the current location $\hat{\boldsymbol{b}}$. Secondly, around the $\hat{\boldsymbol{b}}^-$, segmentation model finds a list of blobs $\{B_j\}$. Trackers in the list

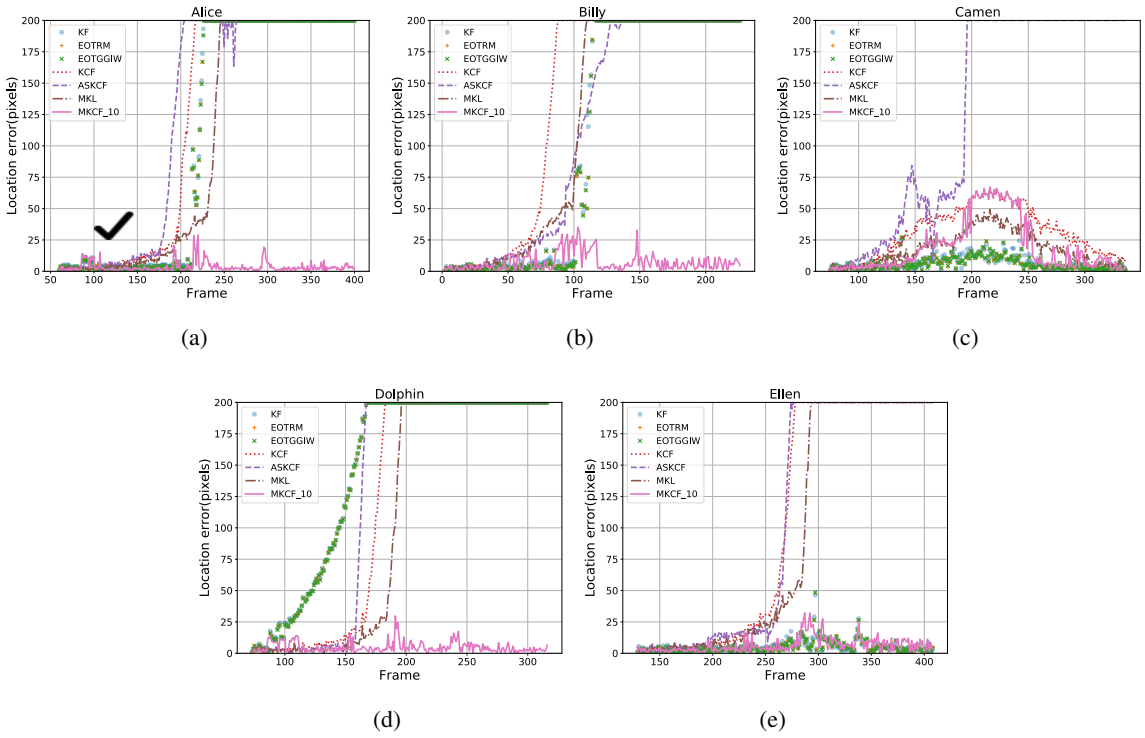


Fig. 14: Center location error frame by frame

votes for the blob which contains the target from the blob list. The bounding rectangle r of the chosen blob is used to initialize a new tracker T_{m+1}. When the container of trackers is full (the number of fused trackers exceeds the limitation N_{th}) and the least PSR score is lower than the voting threshold ($s < S_{th}$), the tracker with the least PSR score would be cleared from the tracker list.

Appendix B

CLE frame by frame

Figure 14 shows the detailed center location error frame by frame for all the trackers in tracking the selected targets. KF and EOTRM are easy to drift away (location error is bigger than 200 pixels suddenly) in tracking Alice (Fig. 14(a)) or Dolphin (Fig. 14(d)), once the segmentation is interrupted by other clutter. KF and EOTRM survive to track Ellen from the start to the end with certain spikes in the Fig. 14 (e) . A spike means that a wrong target association happens,

but recovers in a little while. It reflects the critical state between drifting away and surviving for a tracker. KCF performs bad if the size of target changes sharply and moves fast like Alice and Billy. ASKCF slightly improves the performance of KCF and still suffers to the interruption. The scale of ASKCF is prone to shrink when the tracking PSR drops. MKL gets the second average scores on mean of CLE. Strong features of this kind visual tracker enhance the target association ability during the clutter interruption. However it performs not so well, if the target moving fast with varying intensity. **Our proposed MKCF succeeds to track all the targets robustly with the most smooth curve.**

Appendix C

Highlights

This sub-section is supplementary material only for review, which is not contained in the submitted double-column paper for publishing.

In the Figure 15, we show the tracking results in typical frames. Tracking Alice in Figure 15 (a-c), visual tracker KCF (white) gets lost first due to the fast moving speed and sharp scale changing of Alice in frame #199. ASKCF (gray-blue) struggles to track with shrinking scale while the true target is enlarging. KF (red) and EOTRM (yellow) has motion model. It seems easy for them to follow the quick moving Alice in frame #199. However, when the head-on meeting ship brings interrupted clutter, KF and EOTRM drift away quickly in frame #213 and #234. MKCF (green) succeeds to track Alice from the start to the end.

When tracking Billy in Figure 15 (d-f), KCF and ASKCF already fails before frame #101. We notice the wrong association of KF and EOTRM in frame #102. They jumps to other clutter which is near to the true target. This is also observable in Figure 14 (b), where a spike occurs around the frames #100 to #115. From the frame #122, KF and EOTRM jumps back to the true target. Therefore, they have comparable overall performance in the precision plot (see Figure 10) . The proposed MKCF tracks the Billy smoothly.

Camen in the Figure 15 (g-i) has a big scale changing but maintains a stable local appearance (the tail of the ship) . Visual tracker KCF locates the Camen quite successfully, without considering the scale. KF and EOTRM are taken away by other ship when close meeting happens in frame #191. EOTRM jumps back onto the Camen later, but still suffers to Camen's own multi-path clutter in frame #312.

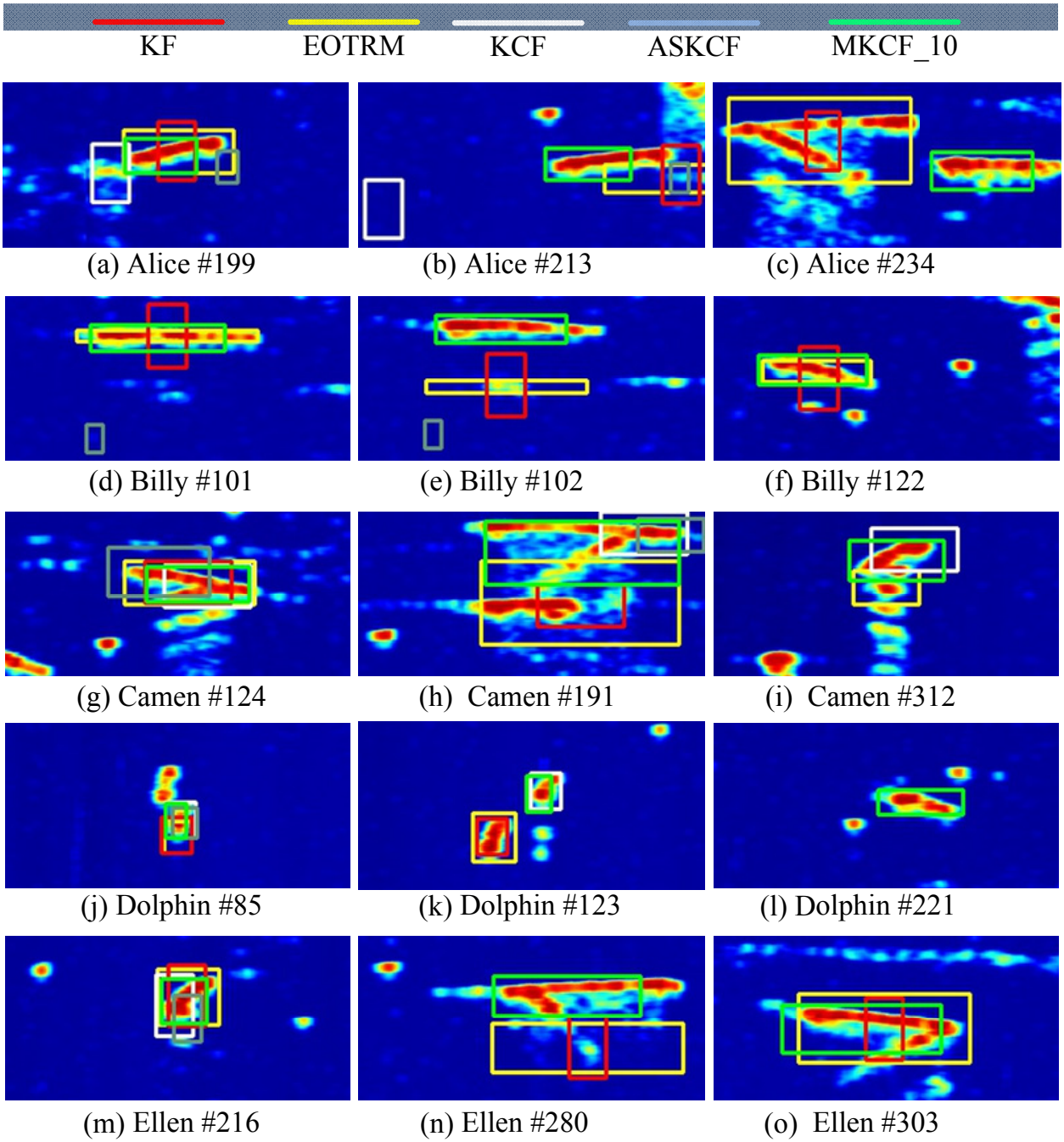


Fig. 15: The highlights of tracking in typical frames. Tracking results of the five trackers are marked in different colors.

Dolphin is overtaking other ship in the Figure 15 (j-l) . KF and EOTRM track the wrong object due to the incorrect association in the frame #123. KCF and ASKCF succeeds to identify Dolphin when ship overtaking happens. However they are not qualified when Dolphin speeds

up and changes scale in frame #221.

Ellen is similar to Billy in moving fast and varying scale sharply. Visual tracker KCF and ASKCF perform badly. KF and EOTRM drifts a little while, due to the heavy clutter of Ellen's multi-path clutter. MKCF tracks the targets more robustly in all cases. It shows the advantages of visual tracker in target association when occlusion or interruption happens. With multiple filters, MKCF improves the robustness of visual tracker in tracking the fast moving object, even they changing the scales quickly.

References

[1] J. Vermaak, N. Ikoma, and S. Godsill, "Sequential monte carlo framework for extended object tracking," in *IEE Proceedings on Radar, Sonar and Navigation*, vol. 152, no. 5, Oct. 2005, pp. 353–363. 5

[2] J. W. Koch, "Bayesian approach to extended object and cluster tracking using random matrices," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 3, pp. 1042–1059, 2008. 5, 8

[3] M. Feldmann, D. Franken, and W. Koch, "Tracking of extended objects and group targets using random matrices," *IEEE transactions on signal processing*, vol. 59, no. 4, pp. 1409–1420, 2011. 5, 8, 32, 33, 34, 38

[4] D. Angelova and L. Mihaylova, "Extended object tracking using monte carlo methods," *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 825–832, Feb. 2008. 5, 8

[5] U. Orguner, "A variational measurement update for extended target tracking with random matrices," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3827–3834, 2012. 5, 8

[6] K. Granström and U. Orguner, "A phd filter for tracking multiple extended targets using random matrices," *IEEE Transactions on Signal Processing*, vol. 60, no. 11, pp. 5657–5671, Nov. 2012. 5, 8

[7] B. Errasti-Alcala and P. Braca, "Track before detect algorithm for tracking extended targets applied to real-world data of x-band marine rada," in *17th International Conference on Information Fusion*, Salamanca, Spain, July 2014. 5, 8

[8] K. Granström, A. Natale, P. Braca, G. Ludeno, and F. Serafino, "Gamma gaussian inverse wishart probability hypothesis density for extended target tracking using x-band marine radar data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 12, pp. 6617–6631, Dec. 2015. 5, 8

[9] G. Vivone and P. Braca, "Joint probabilistic data association tracker for extended target tracking applied to x-band marine radar data," *IEEE Journal of Oceanic Engineering*, vol. 41, no. 4, pp. 1007–1019, Oct. 2016. 5, 8

[10] M. Isard and A. Blake, "Condensation–conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998. 6, 8

[11] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2006, pp. 798 – 805. 6

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of Advances in Neural Information Processing Systems*, vol. 2, 2012, pp. 1097–1105. 6

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248 – 255. 6

[14] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3074–3082. 6, 7, 9

[15] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proceedings of IEEE International Conference on Computer Vision Workshops*, 2015. 6, 9

[16] T. Wang, Y. Chen, M. Zhang, J. Chen, and H. Snoussi, "Internal transfer learning for improving performance in human action recognition for small datasets," *IEEE Access*, vol. 5, pp. 17 627–17 633, 2017. 6

[17] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 983–990. 6, 9, 10

[18] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012. 6, 9, 10

[19] J. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proceedings of the European Conference on Computer Vision*, 2012. 7, 13, 17, 25, 33, 34, 38

[20] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proceedings of the British Machine Vision Conference*, 2014. 7, 9

[21] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *Proceedings of European Conference on Computer Vision*, 2014. 7, 9, 33, 34, 38

[22] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, and L. ehovin et al., "The visual object tracking vot2014 challenge results," in *the Proceedings of the 13th European Conference on Computer Vision*, 2014. 7

[23] M. Tang, B. Yu, F. Zhang, and J. Wang, "High-speed tracking with multi-kernel correlation filters," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 4874–4883. 7, 9, 33, 34, 38

[24] R. Z. Zhang, *Ship navigation radar*. China Communications Press, 1990. 8

[25] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME*, vol. 1, no. 1, pp. 35–45, March 1960. 8

[26] P. R. Kalata, "alpha-belta tracking systems: A survey," in *American Control Conference*, Chicago, IL, USA, June 1992. 8

[27] S.S.Blackman and R.Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House, 1999. 8

[28] Y. Bar-Shalom, P. K. Willett, and X. Tian, *Tracking and Data Fusion: A Handbook of Algorithms*. YBS Publishing, 2011. 8

[29] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, January 2004. 8

[30] K. Granström, M. Baum, and S. Reuter, "Extended object tracking: Introduction, overview and applications," *Journal of Advances in Information Fusion*, vol. 12, no. 2, pp. 139–174, Dec. 2017. 8

[31] E. Maggio and A. Cavallaro, *Video Tracking: Theory and Practice*. Wiley, 2010. 8

[32] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, May 2003. 9

[33] S. Avidan, "Support vector tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1064–1072, 2004. 9, 10

[34] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, "Struck: Structured output tracking with kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2096–2109, Oct. 2016. 9, 10

[35] J. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, March 2015. 9, 10

[36] A. Bibi and B. Ghanem, "Multi-template scale-adaptive kernelized correlation filters," in *Proceedings of the IEEE International Conference on Computer Vision*, February 2015, pp. 613–620. 9

[37] M. Tang and J. Feng, "Multi-kernel correlation filter for visual tracking," in *IEEE International Conference on Computer Vision*, Dec 2015, pp. 3038–3046. 9

[38] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006. 12, 16

[39] R. Rifkin, G. Yeo, and T. Poggio, "Regularized least-squares classification," *In Advances in Learning Theory: Methods, Models and Applications*, pp. 131–154, 2003. 12, 13

[40] B. Schölkopf and A. Smola, *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 12

[41] J. Vermaak, N. D. Lawrence, and P. Pérez, "Variational inference for visual tracking," in *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003, pp. 773–780. 16

[42] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2544–2550. 18

[43] M. E. Liggins, D. L. Hall, and J. Llinas, *Handbook of multisensor data fusion: theory and practice*, 2nd ed. CRC Press, 2008. 23

[44] Y. Zhou, X. Liu, J. Suo, C. Liu, X. Su, and L. Liu, "Nonparametric background model based clutter map for x-band marine radar," in *Proceedings of the 22nd IEEE International Conference on Image Processing*, Quebec City, Canada, September 2015, pp. 123–127. 25

[45] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Addison Wesley, 1991. 25

[46] F. Meyer and S. Beucher, "Morphological segmentation," *Journal of Visual Communication and Image Representation*, vol. 1, no. 1, pp. 21–46, 1990. 26

[47] A. Mordvintsev and A. K., *OpenCV-Python Tutorials Documentation*. https://opencv-python-tutroals.readthedocs.io/en/latest/index.html, 2017, ch. Image Segmentation with Watershed Algorithm, pp. 143–149. 26

[48] K. Granström, M. Fatemi, and L. Svensson, "Gamma gaussian inverse-wishart poisson multi-bernoulli filter for extended target tracking," in *International Conference on Information Fusion*, July 2016, pp. 893–900. 32, 33, 34, 38