2019 International Conference on Computing, Networking and Communications (ICNC): Mobile Computing and Vehicle Communications

FARES: Fast and Accurate Recognition of Exact Scenes on Mobile Devices

Tengpeng Li*, Xiaoqian Zhang*, Teng Wang*, Nam Son Nguyen*, Xiaohui Liang*, and Bo Sheng*
*Department of Computer Science, University of Massachusetts Boston, 100 Morrissey Boulevard, Boston, MA 02125

***Abstract*—Mobile devices represented by smartphones have been continuously evolving to support various applications. In this paper, we study a new mobile application, named *exact scene recognition*, where a user can identify a particular place by comparing two images taken there. This application framework enables users to annotate a scene supporting more descriptive image-based interactions such as mobile augmented-reality applications. We enhance the regular approaches with the assistance of the angle-of-view (AOV) information obtained from the smartphone. Our experimental results show a significant improvement on accuracy compared to the existing solutions.**

I. Introduction

Smartphones and various mobile apps have been becoming an important component in our daily lives. In this paper, we target a new mobile application, where a user can take a picture at a scene or receive the pictures taken by other users at the scene, and when the user visits the scene later, she or he can recognize it through the phone's camera view.

We name our target application function as *exact scene recognition*. It is related to, yet differs from, the traditional scene recognition and object recognition in computer vision that usually classify a scene or object into a known category missing the 'exact' property. In addition, the traditional recognitions often require a large elaborated training set to yield a high accuracy. The computation workload of their common techniques, such as deep learning, may not be suitable for mobile devices. In this paper, our goal is to recognize the exact same scene that has been caught in a pre-loaded image, and there may not be any well recognizable objects.

Our framework FARES is based on feature extraction algorithms in computer vision. Basically, the features of a pre-loaded image and the captured image are compared to determine if they are taken at the same scene. This problem, in our practical setting, is quite challenging when the two images are taken with different angle-of-views (AOV). The existing feature extraction algorithms do not perform well in the comparisons. In our solution FARES, we assume the images are taken by smartphones, and the AOV information is obtained by the on-board sensors. We develop new approaches that can improve the recognition accuracy based on the AOV information. In particular, we propose two new techniques in FARES. The first one considers the ratio of the matching feature points. We use the AOV information to remove the feature points that may be obstructed or greatly altered because of the AOV change. To further improve the accuracy, our second technique is to consider the relative distance between each pair of matching feature points in the two images. The higher the distance is, the less likely the pair of matching feature points are legitimate. Our evaluation is based on experiments with a large set of images, and the results show a significant improvement of the recognition accuracy.

This project was supported by National Science Foundation grant CNS-1527336.

II. Related Work

There have been plenty of researches dedicated to numerous aspects of scene recognition. The existing works could be roughly classified into hand-crafted methods and learning-based methods [1]. For hand-crafted methods, in [2], the feature of neighboring appearance of local descriptors was considered so as to enable their scheme Spatial-LTM to exceed the bag of words model. On the other hand, learning-based methods were used for complex scene recognition, such as Hybrid-CNN in [3], ImageNet-CNN in [4], scale-specific CNN in [5] and Fused DNN in [6]. Finally, there were also researches combining the hand-crafted and learning-based methods ([7] and [8]). All these prior works do not consider the exact scene and usually need a large training set.

Our solution is built upon feature extraction, and there were some widely used existing methods, such as SIFT [9], SURF [10], and ORB [11]. An early successful approach to feature extraction was the Harris Corner Detection [12], which was rotation-invariant. David G. Lowe introduced the Scale-Invariant Feature Transform (SIFT), which enabled the keypoints to be both invariant to the image rotation and scaling [9]. Bay. H presented a speeded-up version of SIFT, named SURF [10]. In [13], the authors proved this fact through an evaluation on the changes of outdoor environment appearances over seasons. In [11], the authors introduced a more enhanced feature extraction method, Oriented FAST and Rotated BRIEF (ORB). It was built on the FAST keypoint detector in [14] and the BRIEF descriptor in [15], where both techniques were well known for their efficient performance and low cost. There were also attempts to complete the vision within the constraints of today's smart phones. In [16] and [17], authors offload the computation to either the edge or nearby devices, which provided useful platforms to compel mobile AR applications.

III. Problem Formulation

We consider an application of scene recognition for mobile phone users. The user pre-loads a set of images each represent-

978-1-5386-9223-3/19/$31.00 ©2019 IEEE

ing a different scene. This set of images can be downloaded from a repository or transferred from other users. In this application, the user holds phone to continuously capture image frames through a camera view. Each frame is examined and compared with the preloaded images for exact scene recognition. In particular, our goal is to exactly recognize the place that one of the preloaded images is taken.

In this paper, we aim to achieve exact scene recognition by analyzing the features of the captured images and the preloaded images. We also consider the additional information the mobile phone provides. Specifically, we assume each image is attached with the information of angle-of-views (AOV) of the camera. Our approach uses the AOV information in feature comparison to improve the accuracy of the application.

Assume a user's phone has loaded a set of n images, $\mathcal{I} = \{I_1, I_2, \cdots, I_n\}$, each I_j with its AOV information A_j. A_j is represented by a tuple of three readings indicating the horizontal, vertical, and rotation angles. To simplify the algorithm description, we only consider the first two angles in this paper. The last rotation angle can be easily incorporated. Let C represent the image captured by the user's phone. The objective of our problem is to search $\mathcal{I}$ and find a matching scene of C, or report none if there is no match.

IV. Design of FARES

In this section, we describe the details of our algorithms. We first briefly introduce the main structure of our algorithm. Then we present the details of three major components.

A. Sketch of the Solution

The main algorithm in FARES (Algorithm 1) enumerates the images in the pre-loaded set $\mathcal{I}$, and compares each of them to the image captured by phone camera (C). Function CompareImages returns the similarity of the two images, and the maximum value is recorded in L_{max}. At the end of the loop, if L_{max} is greater than a threshold, the corresponding image R will be returned as the recognized scene. Otherwise, none of the images matches the captured frame C.

Algorithm 1: Exact Scene Recognition

```
   input : Image set I, Captured frame C
1  for image I_j in I do
2  |  L = CompareImages(C, I_j)
3  |  if L > L_max then
4  |  |  L_max = L, R = I_j
5  if L_max > τ then return R
6  else return null
```

The core technical component in our algorithm is the comparison of the two images. We aim to develop an efficient algorithm that can quickly process the captured images. Our main approach is based on the comparison of the feature points of the two images. In the rest of the paper, we use FP to represent "feature points". The basic intuition of our design is to apply simple but quick comparison of FPs with the assistance of the AOV information to improve the accuracy.

In particular, our algorithm first uses the existing approaches to extract FPs from both images for comparison. Then we apply two new techniques to help improve the accuracy with the assistance of the AOV information. In the first technique, we consider the number of matching FPs between the two images, and measure it as a ratio with the number of the FPs in the pre-loaded image. This ratio, in practice, is quite low with the existing feature extraction algorithms due to the difference of AOVs. We develop a heuristic algorithm to reduce the FPs of the pre-loaded image by eliminating the FPs that might be obstructed or greatly changed because of the different AOV. Our second technique considers the relative distance of a pair of matching FPs in their images. Even if two images have a large number of matching FPs, we use this technique to further avoid the possibility of false positives. Our algorithm calculates the average distance of all pairs of the matching FPs. A higher value of the distance indicates a less likely match.

Algorithm 2 illustrates the main process of comparing two images. This function returns a *score* indicating the likeliness of being the same scene. The score value is between 0 and 1, and the higher the score is, the more likely these two images are taken at the same scene. In Algorithm 2, we first check the matching FP ratio (function call "CalMFPRatio" in line 5), which is a fraction value. If the result is smaller than a threshold τ_r, then the function returns 0 indicating 'no match'. Otherwise, we further examine the relative distances of the matching FP pairs (function call "CalMFPDist" line 9), and return the multiplication of the two fraction values as the *score*.

Algorithm 2: CompareImages(C, I)

```
1  F : Load the pre-processed FPs of I
2  CF : Identify the set of feature points in C
3  A_h, A_v: the AOV differences of the horizontal and
     vertical angles of I and C
4  Calculate the matching FPs,
     MF = { (p_i, p_j) | p_i ∈ F, p_j ∈ CF}
5  r = CalMFPRatio(MF, F, CF, A_h, A_v)
6  if r < τ_r then
7  |  return 0
8  else
9  |  return r · CalMFPDist(MF, F, CF)
```

Specifically, our solution consists of three components: pre-processing of the image set, calculating the matching FP ratio, and calculating the matching FP distance. We present them separately in the following subsections.

B. Pre-Process the Image Set

This step basically serves the second component of calculating the matching FP ratio. Once a new image is received, the user device will process the images to prepare for the exact scene recognition. In this process, we first use the regular feature extraction algorithms such as SIFT [9], SURF [10] and ORB [11]. The user can configure to use a particular algorithm or incorporate new algorithms in our framework.

After obtaining a set of FPs for each I_j, this component builds four "neighbor" lists for each FP. Let FP_j indicate the set of FPs of I_j and p_i denote the i-th FP in FP_j. For each p_i, we create four *neighbor lists* of FPs representing other FPs

in the four directions of p_i, namely 'up', 'down', 'left', and 'right'. For example, the FPs in the 'left' neighbor list are on the left of p_i in the image. In addition, we set another threshold w to confine the FPs on the neighbor lists in a belt-shape region. The definition of 'left' neighbor list is as follows:

$$p_i(left) = \{\ p_j \mid p_j.x < p_i.x \text{ and } |p_j.y - p_i.y| < w\ \},$$

where (x, y) represent the coordinates of the FP. In the image, the coordinates of the top-left corner and bottom-right corner are (0,0) and (width, height) respectively. All the neighbor lists are sorted according to the spatial distance to p_i.

Fig. 1 illustrates an example of the four neighbor lists, each with a different color, of an FP (marked by a big white circle). Some FPs marked by two colors belong to two neighbor lists.

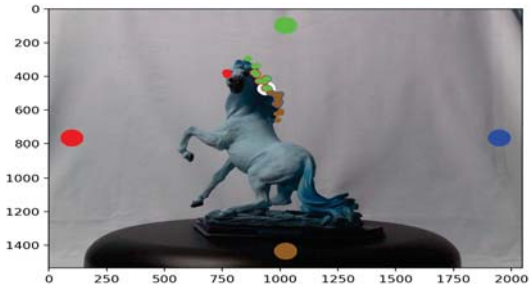


Fig. 1: An example of neighbor lists

The intuition of finding the neighbor lists is to identify the FPs that may form a surface and block the given FP when AOV changes. The details will be explained in the next subsection.

C. Calculate the Matching FP Ratio

In the first step, we consider the ratio of the matched FPs and the total FPs in the pre-loaded image, i.e., $\frac{|MF|}{|F|}$. Intuitively, the higher the ratio is, the more likely the captured image is the same scene as the known image. However, the high accuracy is not guaranteed because same scene may be captured from different angels and different scenes may yield a considerate set of matching FPs.

In this subsection, we present a new algorithm aiming to improve the matching FP ratio of the images taken at the same scene. We observe that with the change of AOV (the details will be illustrated in Section V), the typical algorithms yield low matching ratios. It is because the original FP set F no longer effectively represents the image. Therefore, we apply two approaches to reducing F and generating a more effective FP set. First, some FPs in F may disappear in the captured image's FP set CF because obstruction may happen when the AOV changes. Second, some FPs may still exist but their descriptors have changed so significantly that the matching algorithms do not identify them as the matching FPs.

Algorithm 3 presents the details of our solution. Basically, we try to find a set of FPs that are blocked in the captured image, which should not be counted in calculating the ratio. BF represents this set of 'blocked' FPs. And, eventually we use $|MF|/(|F|-|BF|)$ to represent the matching FP ratio (r_1 in line 12). We use a loop (lines 1–11) to examine each un-matched FP p_i. According to the AOV change, we identify the related neighbor lists of p_i. For example, if the camera view moves to the left side, we will consider the 'left' neighbor list of the given FP because this FP may be blocked by a object on the left side, and this object may be represented by some FPs in the 'left' neighbor list. Next, we scan the sorted neighbor list and find the first FP that is in the matching FP set (line 4). Starting from there, we suppose the area that is blocked has passed. Then we use another loop (lines 5–8) to continue scanning the neighbor list, and count the number of matching FPs (variable c). The scan stops when c is over a threshold T_n. T_n is set to be a small number, e.g., 5, to indicate a reasonable number of FPs that can represent a surface or a portion of an object. Then we check the total number of FPs we have checked during the course of this scan. The ratio α is calculated in lines 9–10. If those matching FPs in the neighbor list are sparsely located, it is not the case we assume to be. Otherwise, if the ratio is higher than a threshold, we add this un-matched FP in the the set BF. Finally, we use r_1 in line 15 to represent the adjusted matching FP ratio.

Algorithm 3: CalMFPRatio(MF, F, CF, A_h, A_v)

1 **for** *an unmatched FP* $p_i \in F \setminus MF$ **do**
2 Based on A_h and A_v, determine the direction change of the AOV, $dc = [left|right|up|down]$
3 $NL = p_i(dc)$ //fetch the pre-computed neighbor list
4 $u = min\{x, NL[x] \in MF\}$ //first matched neighbor
5 **for** $x = u$ *to* $|NL|$ **do**
6 **if** $NL[x] \in MF$ **then**
7 $c \leftarrow c + 1$
8 **if** $c \geq T_n$ **then** break
9 **if** $c < T_n$ **then** $\alpha = \frac{c}{|NL|-u}$
10 **else** $\alpha = \frac{T_n}{x-u+1}$
11 **if** $\alpha \geq T_b$ **then** $BF \leftarrow BF + p_i$
12 $r_1 = \frac{|MF|}{|F|-|BF|}$
13 $r_2 = \frac{90-|A_h|}{90} \cdot \frac{90-|A_v|}{90}$
14 return $r_1 \cdot r_2$

In addition, we consider the FP descriptor information may change due to the AOV change, even if they are not blocked by any other objects. Our algorithm uses a simple heuristic reduction function that is reversely proportional to the angle change. If the angle change is 90°, all the original FPs will be gone. For any change A between 0 and 90, we assume $\frac{90-A}{90}$ portion of visible FPs will be identified as matching FPs (r_2 in line 13). Eventually, $r_1 \cdot r_2$ is returned as the result.

D. Calculate the Matching FP Distance

The last component in FARES is to calculate the relative distance of matching FP pairs. While the previous component calculating the matching FP ratio only considers the number of matching FPs, it is possible for some false or similar images to pass the threshold, especially if the pre-loaded image has a small set of FPs. In this component, we further check the coordinates of the matching FPs in their images.

The details are illustrated in Algorithm 4. We first calculate the centroid points of the matching FPs in both $F \cap MF$ and $CF \cap MF$ (line 1). Then we consider these two centroid

points as the origin in both images, and calculate the relative coordinates of each matching FP (lines 2–4). Next, in lines 5-6, we calculate the spatial distance of each pair of matching FPs according to their adjusted coordinates assuming that two coordination systems are merged by aligning their origins. The average distance D is derived in line 7. Finally, we develop a scoring function on D. When the distance is smaller than a threshold T_d, we assume there is no loss on the score by returning 1. When D exceeds T_d, we use a exponentially decreasing function to represent the penalty on the score (line 11). β is a pivot parameter controlling the curve of the function. In our default setting, $\beta = 1$.

Algorithm 4: CalMFPDist(MF, F, CF)

1 Calculate the centroid point of $F \cap MF$ and $CF \cap MF$, indicated by CP_F and CP_C respectively
2 **for** $p_i \in F$, $p_j \in CF$ **do**
3 $\quad p_i.x \leftarrow p_i.x - CP_F.x$, $p_i.y = p_i.y - CP_F.y$
4 $\quad p_j.x \leftarrow p_j.x - CP_C.x$, $p_j.y = p_j.y - CP_C.y$
5 **for** $(p_i, p_j) \in MF$ **do**
6 $\quad D = D + dist(p_i, p_j)$
7 $D = \frac{D}{|MF|}$
8 **if** $D < T_d$ **then**
9 $\quad$ return 1
10 **else**
11 $\quad$ return $e^{-\frac{D-T_d}{\beta \cdot T_d}}$

Combining the value returned by Algorithm 4 with value of Algorithm 3 in Algorithm 2, this component is proved effective in helping filter the false positives in Section V.

V. Performance Evaluation

In this section, we present our evaluation results.

Implementation and workload. We implement the algorithms with Python3 and OpenCV 3.4.1 [18]. The image set for evaluation are several groups of object images from [19]. For each object group, this library provides the image taken from different angles with an interval of $5°$. We consider the image taken at $0°$ as the pre-loaded image, and the images taken at other angles as the captured images. Due to the page limit, we only evaluate the images with horizontal AOV changes in this paper. In addition, we impose one 'false' image that is not relevant. Fig. 2 shows a subset of a group of our test images.

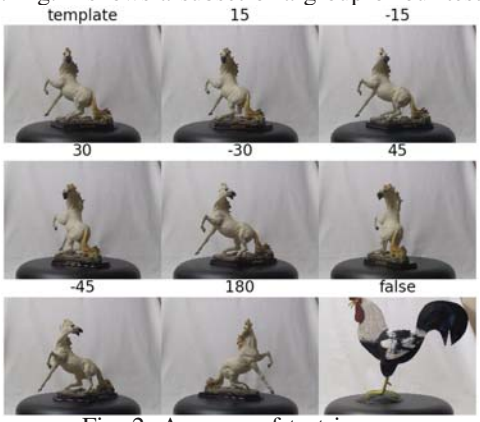


Fig. 2: A group of test images

Our experiments use ORB [11] to detect and describe FPs extracted from each image. To test the effectivity of our method without potential influence from the matcher, we use Brute-Force matcher in the evaluation.

Evaluation of accuracy. Our metric is the accuracy, which is commonly quantified by *precision* and *recall*. Here the precision rate is defined as $TP/(TP+FP)$ and recall rate is defined as $TP/(TP+FN)$, where TP, FP, and FN is true positive, false positive, and false negative respectively.

First, we randomly select 160 images from 7 groups in [19]. If the AOV of the image is between $-90°$ and $90°$, we expect the scene can be recognized. And, we call those images *positive images*. The images whose AOV is out of this range and the irrelevant images we impose are called *negative images* as we expect the recognition algorithms to return negative on them. We apply FARES as well as regular ORB and SURF feature points matching to recognize the scene. In FARES, an image is labeled as positive if the score is higher than 0.3. The other parameters in the experiment are set as follows: $T_b = 0.5, T_n = 5, T_d = 150$. For ORB and SURF alternative, the similarity score is given by $|MF|/|F|$, i.e., the ratio of the number of matching FPs and the number of the original FPs.

Table I compares the accuracy of the three schemes. We observe that ORB and SURF have 100% precision rates, but extremely low recall rates ($< 11\%$). It is mainly because the matching FP ratios are very low in both ORB and SURF, and they only identify a small number of positive images. The results are also reflected by the scores in Table I. For both ORB and SURF, the average scores for positive and negative images are low. FARES outperforms both alternative schemes by yielding a very high recall ratio ($> 97\%$) while keeping the precision sufficiently high (79.67%). It indicates that FARES is able to identify significantly more positive images.

	Precision	Recall	Avg score for positive images	Avg score for negative images
FARES	79.67%	97.02%	0.83	0.36
ORB	100%	10.89%	0.10	0.003
SURF	100%	8.91%	0.13	0.02

TABLE I: Comparison of different algorithms on 160 images

Fig. 3 compares the scores given by FARES, ORB, and SURF for the images taken at several discrete angles between $-45°$ and $45°$. It also shows the average score of the negative images. Apparently, in FARES, all the legitimate images yield high scores. And, we can easily set a cut-off line to distinguish the positive images from the negative images. ORB and SURF, however, both yield low scores in all the tested cases. The scores of some positive images such as $-45°$, $-30°$ and $45°$ are very close to that of negative images.

Above all, FARES is superior to the traditional ORB and SURF in terms of the accuracy.

Impact of matching FP ratio. In addition, we evaluate the major function presented in Algorithm 3 that eliminates the 'blocked' FPs from the original FP set. In Algorithm 3, T_b is an important parameter that determines if an FP needs to be excluded in the ratio calculation. In this experiment, we select three groups of images with a set of fixed angles. In Fig 4 we vary the value of T_b and plotted the scores given by FARES.

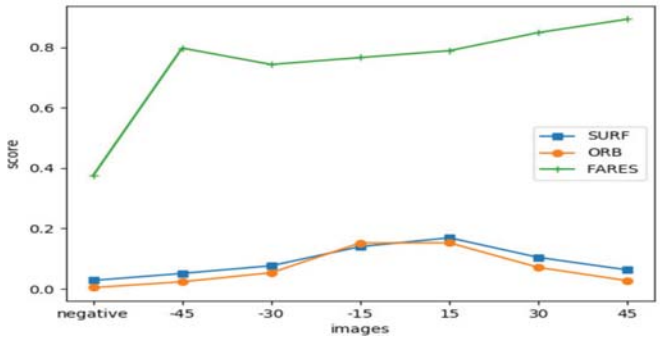


Fig. 3: Comparison of the scores

Again, we test the positive images taken with AOV change in $[-45°, 45°]$, and we impose a false image in each group. In addition, we also test the image with $180°$ AOV change, which is considered as a similar but not the same scene in our evaluation. Because when taken from the opposite side, the image is supposed to hold very few original FPs. In Fig 4, we distinguish the curves of the false and $180°$ images with dashed lines. We observe that there is wide space between the positive and negative image for group 'lamp' and 'clock'. The gap for group 'horse' is narrower, but still considerablely wide for us to set a threshold. And, this property holds for all the values of T_b though a smaller value yields a bigger difference.

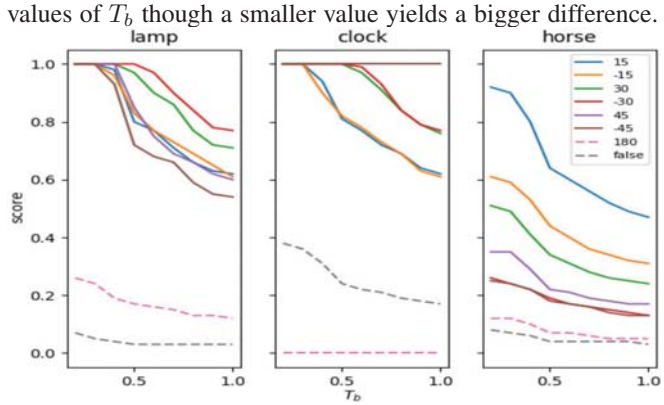


Fig. 4: Scores given by FARES with varying T_b

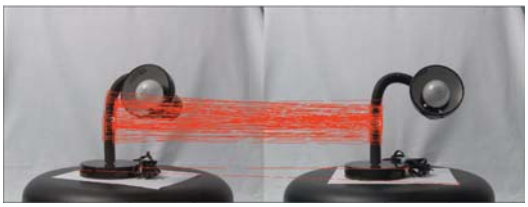
Impact of matching FP distance. Finally, we show the impact of Algorithm 4. Due to the page limit, we use a case study to demonstrate the effectiveness. Fig. 5 shows the matching FPs of a positive image taken at angle $30°$, and false image, with the pre-loaded template image on right side.

As we can tell, the traditional feature extraction algorithms actually identify many matching FPs in the false image. However, these FPs' coordinates do not match their counterparts in the original figure. As a comparison, the matching FPs in Fig. 5a are quite consistent. Our Algorithm 4 certainly gives a high score for Fig. 5a, and very low score for Fig. 5b.

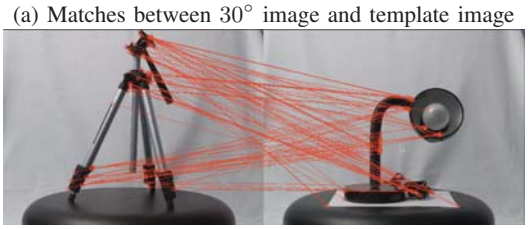
In summary, FARES can accurately recognize the exact scene by comparing two images taken with different AOVs. Both Algorithm 3 and Algorithm 4 are effective in improving the accuracy of the traditional feature extraction algorithms.

VI. Conclusion

This paper targets on an application of exact scene recognition. The main problem is to compare two images taken at the same scene but with different angle of views. Our solution is built on the feature extraction algorithms, and we present two new techniques to identify 'blocked' FPs and consider the spatial distance between each pair of matching FPs. The experimental results show that our solution significantly improves the accuracy.



(a) Matches between $30°$ image and template image



(b) Matches between false image and template image

Fig. 5: Plot of matching FPs in two cases

References

[1] X. Cheng, J. Lu, J. Feng, B. Yuan, and J. Zhou, "Scene recognition with objectness," *Pattern Recognition*, 2018.
[2] L. Cao and L. Fei-Fei, "Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes," in *ICCV*, 2007.
[3] G.-S. Xie, X.-Y. Zhang, S. Yan, and C.-L. Liu, "Hybrid cnn and dictionary-based models for scene recognition and domain adaptation," *TCSVT*, 2017.
[4] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Adv Neural Inf Process Syst*, 2014.
[5] L. Herranz, S. Jiang, and X. Li, "Scene recognition with cnns: Objects, scales and dataset bias," in *CVPR*, 2016.
[6] X. Du, M. El-Khamy, J. Lee, and L. Davis, "Fused dnn: A deep neural network fusion approach to fast and robust pedestrian detection," in *WACV*, 2017.
[7] L. Jin, S. Gao, Z. Li, and J. Tang, "Hand-crafted features or machine learnt features? together they improve rgb-d object recognition," in *Multimedia (ISM)*, 2014.
[8] W. Li, S. Manivannan, S. Akbar, J. Zhang, E. Trucco, and S. J. McKenna, "Gland segmentation in colon histology images using hand-crafted features and convolutional neural networks," in *ISBI*, 2016.
[9] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc IEEE Int Conf Comput Vis*. IEEE Computer Society, 1999.
[10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, 2008.
[11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proc IEEE Int Conf Comput Vis*, 2011.
[12] C. Harris and M. Stephens, "A combined corner and edge detector." in *AVC*, 1988.
[13] C. Valgren and A. J. Lilienthal, "Sift, surf & seasons: Appearance-based long-term localization in outdoor environments," *Rob Auton Syst*, 2010.
[14] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *ECCV*, 2006.
[15] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *ECCV*, 2010.
[16] P. Jain, J. Manweiler, and R. Roy Choudhury, "Overlay: Practical mobile augmented reality," in *MobiSys*, 2015.
[17] H. H. Harvey, Y. Mao, Y. Hou, and B. Sheng, "Edos: Edge assisted offloading system for mobile devices," in *ICCCN*, 2017.
[18] G. Bradski, "The OpenCV Library," *DDJ*, 2000.
[19] P. Moreels and P. Perona, "Evaluation of features detectors and descriptors based on 3d objects," *IJCN*, 2007.