EMOVIS - An Efficient Mobile Visual Search System for Landmark Recognition

Dawei Li
Computer Science and Engineering Department
Lehigh University
Email: dal312@lehigh.edu

Mooi Choo Chuah
Computer Science and Engineering Department
Lehigh University
Email: chuah@cse.lehigh.edu

***Abstract*—Traditionally, content-based image retrieval systems (CBIR) are designed to allow users to search for images in large databases which match closely with users' query images. Recent emergence of powerful mobile devices equipped with digital cameras have led to the emergence of several interesting mobile CBIR applications. Due to the limited resources in mobile devices, it is critical that the image matching engine within any mobile CBIR system be efficiently designed. Many existing image matching engines use SURF-based methods which return many keypoints, and hence are not quite suitable for mobile devices. In this paper, we present an efficient mobile visual search system (EMOVIS) which allows mobile users to retrieve relevant information using image-based queries. EMOVIS uses two unique salient keypoint identification schemes we designed which allow image matching to be conducted efficiently and with high accuracy. In addition, EMOVIS includes an image cropping scheme which eliminates irrelevant regions within a query image. Such cropping minimizes query latency, bandwidth usage and the energy cost of using EMOVIS. Via extensive evaluations using ZuBuD dataset and our own image dataset, we showed that EMOVIS can achieve higher than 92% accuracy with low computational and energy cost.** [1]

***Index Terms*—mobile visual search, CBIR, SURF, salient keypoints**

I. Introduction

In recent years, powerful mobile devices with one or two embedded cameras e.g. IPhone or Android-based phones have emerged and become pervasive. New mobile media search related applications have appeared. Among these, mobile location search [1], [2] is one of the most popular mobile media search applications. Users can search information about locations, landmarks or products by just taking a photograph of the object of interest using their mobile devices. Each captured image is sent as a query over wireless networks e.g. cellular or WiFi networks to a remote server. The server then compares the query image with training images of recognizable objects or landmarks stored in an image database in an attempt to identify the object (e.g. a landmark or a product) within the query image.

In addition, some emerging augmented reality (AR) applications also provide mobile visual search. A tourist who visits a place he/she is unfamiliar with can conduct a virtual tour of his/her new environment using an AR-based tourist guide system. That tourist can also snatch a photo of any building he/she encounters and retrieve interesting information e.g. history of the building via an image-based query.

All the above applications described above are built using an image matching engine. The success of such an engine relies on whether salient keypoint descriptors can be identified which (i) allows images of different targets to be distinguished, (ii) endures content variability among training images of the same target and (iii) endures possible distortions added to a query image (e.g. inclusion of trees in a building image) during the mobile image capture process. Often, incorrect locations with visually similar appearances are returned as top matches when the image matching engine failed to identify a good match. Such poor performance is consistent with our own experience using Google image search feature as well as results reported in other state-of-the-art systems [2], [3], [4], [5]. Furthermore, many existing image matching engines use SURF-based methods [20]. However, such methods usually involve many keypoints, and hence are not quite suitable for mobile devices.

In this paper, we focus on designing an efficient mobile visual search system for landmark recognition called EMOVIS. To improve the computational efficiency for image matching, we design two novel algorithms to extract the most salient keypoints of a landmark from a set of training images. All extracted salient keypoint descriptors are later used by EMOVIS to construct a kd-tree to speed up the image matching process.

To further improve the matching accuracy and reduce both the matching processing and energy cost, we also design an image preprocessing scheme. Before being sent to a remote EMOVIS server, a query image is first resized to a reasonable small scale according to its original aspect ratio. After resizing, the query image is cropped to remove irrelevant regions e.g. trees, which make the image less distinctive. Such a preprocessing technique leads to higher matching accuracy and improved query response time.

We conduct extensive studies using a prototype EMOVIS system we developed to compare our approach with two existing approaches, namely (a) a basic hessian-value based approach [14] and (b) an enhanced hierarchical grid based detector [9]. Our results indicate that EMOVIS can achieve a high matching accuracy ($> 92\%$) with faster query response time when compared to these two existing approaches. In addition, we report on the energy consumption of using EMOVIS on Android-based devices.

In summary, our efficient mobile visual search (EMOVIS) system for landmarks has the following unique features:

- uses MaxUnion and MaxMin Salient Keypoint Identification schemes designed to identify salient keypoints from a set of training images of each landmark during an offline training phase,
- achieves fast image-based query response time using a visual-feature based kd-tree constructed from all salient

[1]This project is supported by NSF NeTS Grant 1049845.

keypoints,

- image-preprocessing scheme which removes irrelevant regions in a query image such that matching accuracy and energy cost can be improved, and
- an image matching engine which processes image-based queries asynchronously to achieve higher query throughput.

The rest of the paper is organized as follows: Section II will discuss related work. We present the system architecture of EMOVIS and our design considerations in Section III. In section IV, we present the detailed system design. Then, we report the evaluation results in Section V. We conclude in Section VI by discussing some near future work which we would like to explore.

II. Related Work

A critical component in Content-Based Image Retrieval Systems (CBIR) is the image matching engine. Several popular methods have been designed for image matching e.g. SIFT detector which is robust in scale, orientation, and viewpoint variations. However, the robust features such as SIFT or SURF are time consuming and hence not suitable for mobile devices. Several simplified-version of these features are designed to achieve faster speed. For example, the authors in [6] present a modified SIFT created from a fixed patch size of 15x15 pixels and use only a 36-dimension descriptor. The authors in [7] combine the simplified SIFT with a scalable vocabulary tree to achieve interactive object recognition on mobile phones. The simplified features incur less computational cost which is required for mobile applications. Unfortunately, while such methods can speed up the matching process, they sacrifice the robustness feature. In addition, such methods can only improve the matching speed by 2-3 times when compared to the traditional more robust SIFT or SURF-based detectors.

The authors in [9] proposed a hierarchical image partitioning method where a query image is divided into 4x4 grids and these grids are used to match against training images in their image database. They assume hierarchical partitioning is also performed on every image in their database. There are a total of 16 (1x1 grids)+9 (2x2 grids)+4 (3x3 grids) and 1 (whole image)=30 hierarchical grids for one image and any grid must contain at least N_o features in order to be stored as part of the relevant grids for that image. Their approach however incurs too much storage and memory cost since it needs to maintain information of many hierarchical grids per image. In [14], the authors constructed a kd-forest using keypoints identified from all training images in the database. Additional trees are constructed as new training images are added to the database. Using the ZuBuD database, this approach can yield 92% accuracy with 300 keypoints from each training image. Their average matching time is 300ms. Our approach can achieve such accuracy with only 100 salient keypoints per landmark and incurs only an average matching time of 95ms.

III. System Overview

A. System Architecture

As in several previous works [10][11], we adopt a client-server system architecture. Figure 1 illustrates the system architecture of our efficient mobile visual search system (EMOVIS) and the basic data flow of our system. Our system consists of client software running on mobile devices equipped with cameras and a remote server which can be hosted in a cloud. Content and image databases are attached to the server. The image database houses training images of all buildings (or landmarks) which EMOVIS recognizes, and the salient keypoint descriptors of different faces of these recognized buildings/landmarks. Each building has a unique object identifier which can be used to extract information relevant to that building e.g. history or design architect of a building from the content database.

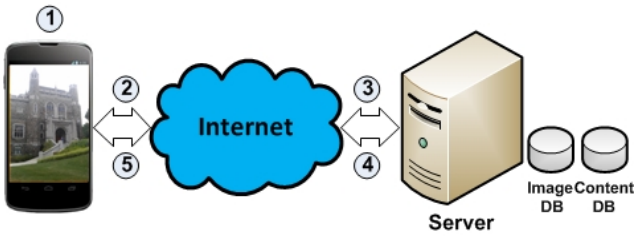

Fig. 1: System Architecture. (1) A user takes a photo with the phone camera and the photo is then preprocessed by EMOVIS client software running on his mobile device. (2) The preprocessed image is then sent to a remote server via a wireless connection. (3) The remote server matches the query image against all trained images in the image database, and retrieves the information relevant to that matched landmark once it is identified. (4) The server returns the retrieved landmark content to the user's device. (5) Our client software then displays the received contents on the screen.

The client-server architecture is chosen because the intensive image matching process should be conducted at a remote high-end server rather than locally in the mobile devices due to their limited battery and computing resources. Furthermore, mobile devices typically do not have enough storage space for all the training images.

B. System Design Considerations

To design an efficient mobile CBIR system, we need to make two major design decisions, namely, (i) tradeoff between matching accuracy and matching speed, (ii) tradeoff between image quality, bandwidth usage, and energy consumption.

1) Tradeoff between Image Matching Accuracy and Image Matching Speed: The default SURF algorithm usually detects more keypoints than are necessary for achieving certain matching accuracy [12][17]. To speed up the matching process, the system must prune the set of keypoints to retain only those important ones. Fewer keypoints means faster matching processing speed but it may affect the matching accuracy when insufficient number of keypoints are used. Therefore, the system must rely on a keypoint pruning algorithm which produces a minimal set of salient keypoints that will yield the required matching accuracy.

2) Tradeoff between Query Image Quality and Network Delay: The query image must be of sufficient quality so that a sufficient number of keypoints can be extracted to match with the salient keypoints of all trained landmarks stored in the image database. However, better image quality means a larger file size which translates into higher energy cost and longer network delay when it is being sent over a wireless link. Hence, the client software must pre-process the query image to an appropriate size before sending it to the remote server.

Overall, our goal is to design a robust mobile visual search system such that it provides satisfactory user experience which means we can sacrifice a little bit of matching accuracy to achieve shorter query latency and lower energy cost. The system should be tunable to meet the minimum required matching accuracy at the lowest possible energy cost and query latency.

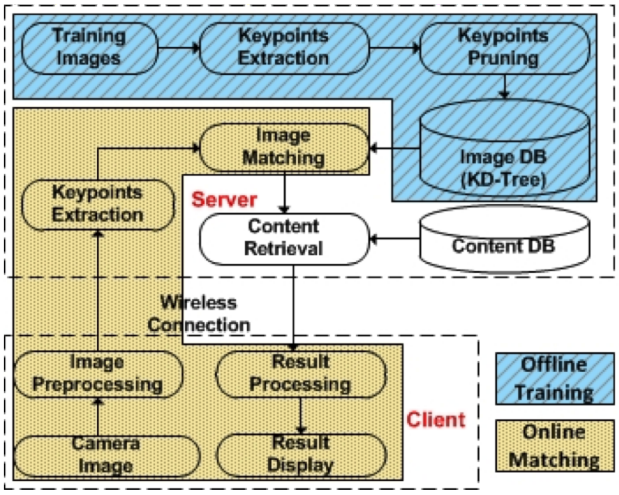

Fig. 2: Software Architecture of EMOVIS

IV. Detailed System Design

Figure 2 illustrates the software architecture of EMOVIS, and the data flow of its operations. The server includes an offline image training module and an online image matching module. The image training module is responsible for identifying salient keypoints of every landmark. In EMOVIS, each face of a building is considered a separate landmark object. The keypoint extraction submodule extracts a keypoint set from each training image. The keypoint sets from all training images are then fed into the keypoint pruning submodule to identify salient keypoints which can be used to uniquely identify a particular landmark object. The identified salient keypoints of all landmarks are then stored in the image database and each landmark object is given a unique identifier. The image matching module extracts keypoints from a query image which are then sent to the image matching submodule for landmark recognition. Once a matched landmark is identified, the server can then use its unique landmark object identifier to extract content about this landmark from the content database. Information related to that landmark e.g. the history of that building, is stored in the content database. The content description of the identified landmark can then be sent to the querying user.

The client software basically performs the image preprocessing function before a query image is sent to the remote server. The image preprocessing involves two tasks. The first task is to resize the image into one with the appropriate quality. The second task is to detect and remove irrelevant parts of a query image such that any potential interference from these irrelevant parts to the matching process can be reduced. Since our system is targeting buildings or landmarks, we focus on removing green plants (e.g. trees, grass) from the query image. Such image cropping allows EMOVIS to achieve higher matching accuracy and lower computational cost. The cropped image is then compressed and sent over a wireless network to the remote server. We choose to send a compressed image rather than the descriptors extracted from the cropped image because the total size of keypoint descriptors can be three times larger [9][14] than the size of a compressed image, and hence sending the compressed image incurs less bandwidth and energy cost. Upon receiving a query response from the server, the client software will display the matched result on the screen of a user's mobile device.

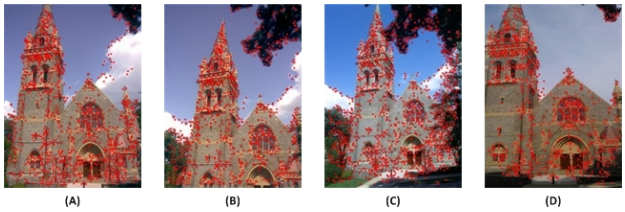

Fig. 3: Keypoints Identified from 4 Training Images using existing SURF detector

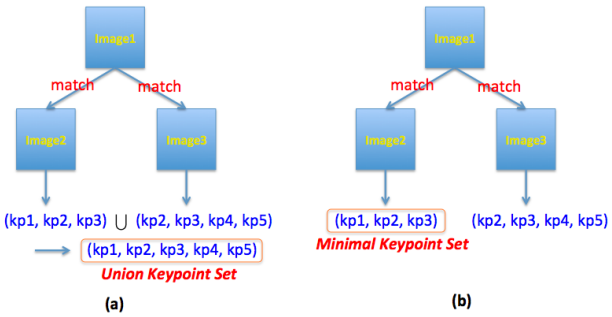

Fig. 4: *Union Keypoint Set* and *Minimal Keypoint Set*

A. Keypoint Pruning for Training Images

A typical SURF-based keypoint extractor can produce many keypoints from each image. In Fig 3, we showed the keypoints obtained using the SURF detector from four training images of Memorial Church in Lehigh University. It is obvious from Fig 3 that many keypoints are returned using the existing SURF detector, and some of them do not represent the building, e.g. the keypoints on the trees and the cloud. Hence, for fast and accurate matching operations, we need to design efficient algorithms to identify a minimal set of keypoints that can be used to uniquely identify a landmark object.

In this work, we design two schemes for identifying salient keypoints from a set of training images for a landmark. To better explain our schemes, we first give two important definitions:

Definition 1: the *Union Keypoint Set* of a training image is the union of all keypoint sets produced by pairwise matching this image with other training images of the same landmark (as shown in Figure 4 **(a)**).

Definition 2: the *Minimal Keypoint Set* of a training image is the smallest keypoint set produced by pairwise matching this image with other training images of the same landmark (as shown in Figure 4 **(b)**).

In both schemes, a *Union Keypoint Set* (**UKS**) is constructed for each training image. The first scheme, the MaxUnion-Based Salient KeyPoint Identification (MaxUnion-SKI) scheme, finds the *Best Training Image* which produces the largest **UKS**. The second scheme, the MaxMin-Based Salient KeyPoint Identification (MaxMin-SKI) scheme, finds the *Best Training Image* that produces the largest *Minimal Keypoint Set*. For both schemes, the **UKS** associated with the *Best Training Image* will then be used as the Master Salient Keypoints (***MSK***) of that landmark.

The pseudo code for the *MaxUnion-Based Salient Keypoint Identification* (MaxUnion-SKI) method is shown in **Algorithm 1**. It first matches the keypoints extracted from a training image with those extracted from another training image. The matched keypoints from the training image (expressed as kps) are "union"ed with the existing union set of matched keypoints found so far and relevant counters are updated before the updated union set is stored in **UKS**. The training image which

Algorithm 1 MaxUnion-Based Salient Keypoint Identification (MaxUnion-SKI) Method

Input: A set of training images $S_{Img} = \{Img_1, Img_2, ..., Img_K\}$
Output: ***Master Salient Keypoints*** MSK

```
master ← null
maxCount ← 0
for each image Img_i ∈ S do
    UKS← ∅                                   # Union Keypoint Set
    for each image Img_i ∈ S_img do
        if Img_i ≠ Img_j then
            kps ← MatchKeypoint(Img_i, Img_i)
            UKS← Union(UKS, kps)
        end if
    end for
    if len(UKS) > maxCount then
        maxCount ← len(UKS)
        MSK ← UKS
    end if
end for
return MSK
```

returns the largest **UKS** will be the *Best Training Image* and that largest **UKS** will be used as the ***MSK*** for that landmark.

A drawback of this method is that when there are a few extremely similar images (e.g. two continuously captured photos), one of them is quite likely to be selected as the *Best Training Image*. However, if those extremely similar images are taken from a bad angle or at a distance, they do not uniquely represent that landmark and hence these keypoints should not have been included in ***MSK***.

The *MaxMin Salient Keypoint Identification* (MaxMin-SKI) method (**Algorithm 2**) avoids this drawback. For each training image, the MaxMin-SKI method determines its *Minimal Keypoint Set*. Then, the MaxMin-SKI method chooses the *Best Training Image* which produces the largest *Minimal Keypoint Set*. The *Union Keypoint Set* associated with this *Best Training Image* will be used as the ***MSK*** for that particular landmark object.

Algorithm 2 MaxMin-Based Salient Keypoint Identification

Input: A set of training images $S_{Img} = \{Img_1, Img_2, ..., Img_n\}$
Output: ***Master Salient Keypoints*** MSK

```
master ← null
maxCount ← 0
for each image Img_i ∈ S do
    minCount ← ∞
    UKS← ∅                                   # Union Keypoint Set
    for each image Img_j ∈ S do
        if Img_j ≠ Img_i then
            kps ← MatchKeypoint(Img_j, Img_i)
            UKS← Union(UKS, kps)
            if len(kps) < minCount then
                minCount ← len(kps)
            end if
        end if
    end for
    if minCount > maxCount then
        maxCount ← minCount
        MSK ← UKS
    end if
end for
return MSK
```

Instead of using all the matched keypoints in ***MSK***, we select the *Top N* salient keypoints per landmark object. Recall that our Union function has a counter value associated with each keypoint in ***MSK***. This counter value measures the number of times that keypoint has matched with keypoints from other training images. Thus, the matched keypoints in ***MSK*** can be easily ranked to identify the *Top N* keypoints. Among the matched keypoints with the same counter values, we can further rank them based on other factors e.g. its hessian value [14] or its distance to other higher ranked keypoints such that we can include keypoints that are not close to one another [17]. The *Top N* keypoints will be used as the salient keypoints representing the landmark. In case the number of keypoints in ***MSK*** is fewer than *N*, the remaining keypoints from the *Best Training Image* which are not in ***MSK*** are ranked (e.g. using Hessian Value) and chosen to make up the Top N salient keypoints.

Fig. 5: Top 100 Salient Keypoints Identified by *MaxUnion-SKI and MaxMin-SKI* in the Best Training Images ((B) and (A)) in Fig. 3

In Fig 5, we showed the salient keypoints identified using our two methods. Compared to Fig3, it is obvious that our two methods significantly reduce the number of keypoints that are used to uniquely describe this particular landmark object. We will show in the evaluation section that the matching accuracy obtained using these two methods is high.

B. Image Matching

Some existing matching approaches use a brute-force matching method i.e. a query image is matched against each of the images in a database, and the image with the most matched keypoints is returned as the matched landmark. However, with a large database, such method is extremely slow and not scalable. Therefore, EMOVIS uses a kd-tree based image matching method, i.e. we look for a nearest neighbor for each keypoint descriptor from the query image. After salient keypoints of all landmark objects are identified, these salient keypoints are used to construct a kd-tree. The nearest neighbor search in kd-tree is an O***(log(n))*** algorithm. Suppose we identify the top ***N*** keypoints for each landmark, and there are ***m*** landmark objects, ***k*** keypoints are detected from a query image. The computational complexity for image matching in a kd-tree is O***(klog(mN))*** assuming that there is sufficient system memory.

We use a 2-stage matching method similar to [14] to match a query image: (1) the keypoints extracted from a query image is matched using the constructed kd-tree(s) to identify the top 10 candidate images, and (2) the pair-wise RANSAC-based match is later used to match only these top 10 candidate images to find the best matched image. Although our query matching process is similar to [14], our method is significantly different since (a) in [14], their kd-tree is constructed using top N keypoints from every training image and hence their kd-tree is bigger, and (b) they use Hessian values to rank keypoints while we identify salient keypoints.

C. Query Image Preprocessing

In general, the total size of SURF-based keypoint descriptors can sometimes be three times the size of the original query image [9][14]. Thus, to be more energy-efficient, our EMOVIS client software sends a compressed version of a cropped query image to our EMOVIS server. This cropped image is produced by a query image preprocessing process.

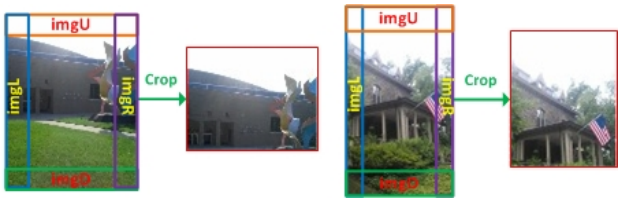

Fig. 6: Two example images processed by *Image Cropping*

The preprocessing process involves two tasks: *Image Resizing* and *Image Cropping*. *Image resizing* has been used in previous work [9] to resize the query image into a fixed 640*480 small-size image. In this work, instead of using a fixed size, we scale a query image down based on its original aspect ratio such that the length of the cropped query image is a preconfigured value (set to 640 in this work). Thus, we maintain the original aspect ratio without causing any image distortion.

The second task of *Image Cropping* is carried out to minimize potential interference caused by irrelevant parts of an image e.g. the green scenery around a landmark during the matching process. The *Green Cropping* task in the EMOVIS client software will remove the green scenery caused by trees, bushes, grass from an image. Our *Green Cropping* algorithm (**Algorithm 3**) is a iterative RGB-based method which identifies a certain region in an image which likely contains green scenery. The MaxRem function returns (i) the maximum intensity among the IntL, IntR, IntU and IntD values extracted from each cropping region of a query image, and (ii) RemIndx identifies the region of an image (L,R,U,orD) that achieves this maximum green intensity value and will be removed. If we cannot find any region to remove, we repeat calling the Green Cropping function one more time using $n = 16$.

Algorithm 3 Green Cropping(Img_q,n=8)

Input: A query RGB-color image Img_q, n
Output: The cropped image, Img_{crop}
1: $Img_crop = Img_q$
2: $imgL \leftarrow$ the left $1/n$ part of Img_q
3: $imgR \leftarrow$ the right $1/n$ part of Img_q
4: $imgU \leftarrow$ the upper $1/n$ part of Img_q
5: $imgD \leftarrow$ the bottom $1/n$ part of Img_q
6: $IntL \leftarrow$ `CheckGreen(`$imgL$`)`
7: $IntR \leftarrow$ `CheckGreen(`$imgR$`)`
8: $IntU \leftarrow$ `CheckGreen(`$imgU$`)`
9: $IntD \leftarrow$ `CheckGreen(`$imgD$`)`
10: $(IntMax, RemIndx) \leftarrow$ MaxRem(IntL, IntR, IntU, IntD)
11: **if** $IntMax > 0.1$ **then**
12: $Img_{crop} \leftarrow$ Remove($Img_q, RemIndx$)
13: **return** `GreenCrop(`Img_{crop}`)`
14: **else**
15: Green Cropping(Img_q,16)
16: **end if**
17: **return** Img_crop

Algorithm 4 Check Green Intensity: CheckGreen(Img)

Input: An RGB-color image Img
Output: The green intensity $gInt$
1: $gValue \leftarrow 0$ # Total green value
2: $rValue \leftarrow 0$ # Total red value
3: $bValue \leftarrow 0$ # Total blue value
4: **for** 100 random RGB pixels $\{p_1, p_2, ..., p_n\} \in image$ **do**
5: $gValue \leftarrow gValue + p_i$`.GreenValue()`
6: $rValue \leftarrow rValue + p_i$`.RedValue()`
7: $bValue \leftarrow bValue + p_i$`.BlueValue()`
8: **end for**
9: $gInt \leftarrow (gValue -$`Max(`$rValue, bValue$`)`$)/gValue$
10: **return** $gInt$

The *Green Cropping* operation reduces the computing burden required at the server during the matching process but we also need to make sure that the additional computing requirement at the mobile device for such cropping is kept to a minimal. Our *Check Green* function in **Algorithm 4** is designed to incur minimal computing requirement. Instead of checking every pixel of a rectangle area (e.g. 38,400 pixels for 640*60 rectangular area), only 100 randomly selected pixels are used and this results in a speed up of 384 times. Figure 6 shows the results of two Lehigh University buildings cropped using our *Green Cropping* algorithm.

V. System Evaluation

In this section, we present evaluation results of our prototype system. We implemented both the client and server modules of EMOVIS using python and OPENCV libraries. Then, we conduct several experiments to measure the performance of both our client and server application. We evaluated the image matching accuracy, server processing time, and server throughput of EMOVIS. In addition, we evaluate the latency required for image pre-processing and its energy cost on an Android-based smartphone.

A. Prototype Implementation and Image Datasets

We built our prototype using OpenCV libraries (version 2.4.5). Our server is configured with Intel(R) Core(TM) i7-2600 CPU @3.40GHz, 16GB memory and 1 Gb Ethernet connection to Lehigh University Network.

In our experiments, two datasets are used: Zurich Building Database (ZuBuD)[13] and Lehigh Building Database (LeBuD). ZuBuD is a popular landmark image dataset used in many previous works [10][14][16]. In ZuBuD, there are 201 buildings each having 5 training images (1005 training images in total) and 115 test images each of which can be matched to one of the 201 buildings. The size for all training images is 640*480 while the size for all testing images is 320*240. We use ZuBuD as the dataset for server evaluation since it contains images of many buildings. In our prototype system, since each face of a building is considered as a separate landmark object, we identify 31 of the ZuBuD buildings which have images of two different faces and therefore we have a total of 232 landmark objects for ZuBuD. The query images in ZuBuD seldom have green scenery so we did not apply the Green Cropping algorithm to these images.

We use LebBuD as the dataset for evaluating our client software. LeBuD is a relatively small image dataset with only 12 buildings and 6 of them have two faces resulting in a total of 18 landmark objects. Each landmark object has 5 training images and 3 testing images which were taken from different angles, distances and with "Green Scenery" contents. The size of each training image in LeBuD is 640*480 while each query image size is not fixed to mimic the various image sizes that different EMOVIS users may send.

B. Matching Accuracy and Query Latency

TABLE I: ZuBuD Image Keypoint Statistical Characteristic

Avg	Max	Min	Std
1276	2654	486	391.47

We evaluate the performance of our prototype system by measuring the matching accuracy and the server processing time. We compare our ***salient keypoint*** based schemes with the pure hessian-threshold keypoint pruning method described in [14][15]. For this set of experiments, we only use the ZuBuD image set as it has a relatively large number of landmark objects and there are significant differences among results of

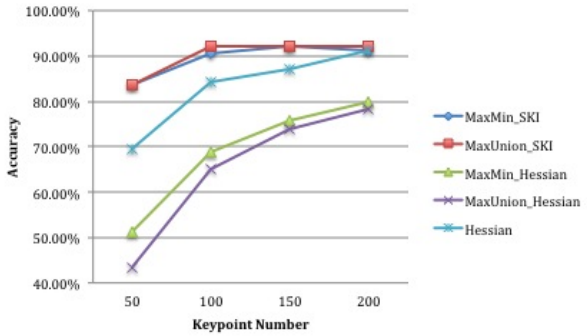

Fig. 7: Matching Accuracy Comparison

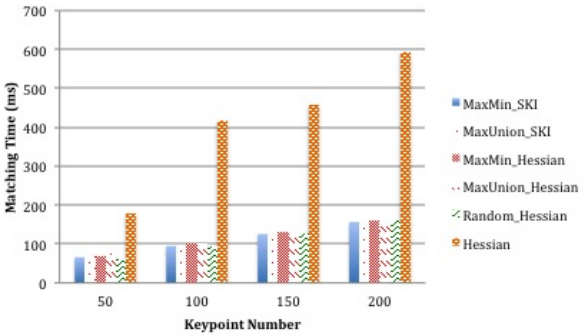

Fig. 8: Matching Time Comparison

different methods. The default Hessian threshold for SURF method is set to 500 and each keypoint has a descriptor of 128 values. Table I shows the statistical characteristic of keypoints extracted from the ZuBuD image set.

We first conduct an experiment using recommended values of the SURF method to estimate its matching accuracy and query response time using such default values. The matching accuracy is defined as the percentage of query images which are recognized correctly. The matching time per query image is defined as the time it takes for an EMOVIS server to return a matching result. For this baseline experiment, a kd-tree is constructed using keypoints extracted from all training images. Then, we extract keypoints from each query image and see which matched landmark is returned. With 115 query images, the matching accuracy using these default values is 94.78%, and the average matching time per query image is 3.94 seconds.

Next, we compare our salient keypoint identification methods with simple hessian-value based keypoint selection [14]. The method used in [14] (referred to as the "Hessian" method) selects the top N hessian-value based keypoints from each training image and constructs a kd-forest using all these extracted keypoints. This method requires a long matching time. Thus, we also tried two related methods, namely (a) "MaxMin_Hessian" and (b) "MaxUnion_Hessian" methods which can speed up the matching process. The "MaxMin_Hessian" is an enhanced Hessian method which uses the top N hessian-value based keypoints only from the *Best Training Image* of a landmark object identified using our MaxMin method. The "MaxUnion_Hessian" is an enhanced method which uses the top N hessian-value based keypoints only from the *Best Training Image* of a landmark object identified using our MaxUnion method.

Figure 7 and Figure 8 show the results of the matching accuracy and the average incurred matching time of each method. The results show that with our proposed methods, the matching accuracy is around 92% while the average matching time is as low as 95ms when Top 100 salient keypoints are used. This is better than the basic "Hessian" method with an accuracy lower than 90% when fewer than 200 keypoints are selected but incurs significantly longer matching time. To achieve an accuracy higher than 90% using the basic "Hessian" method, it requires an average of about 600 ms matching time. The modified hessian-value based keypoint selection methods (MaxUnion_Hessian and MaxMin_Hessian) each has a relatively poor matching accuracy (always lower than 80%) performance.

Next, we compare our approach with the hierarchical grid-based method described in [9]. In this method, each training image is divided into 4x4 grids, and keypoints are extracted from 30 combinations of grids e.g. 1x1, 2x2 grids. Only grids with keypoints higher than a certain threshold (referred to as the retaining threshold) are retained. This method consumes much memory so we did a few minor adjustments so that we can run it to compare with our method. For example, we set the retaining threshold to be 50 because we use query images of smaller sizes. The inlier matched threshold is set to 10 instead of 20. Top 10 candidate images are identified using the kd-tree search rather than using a single returned result.

We ran experiments using 115 test images from the ZuBuD dataset with these three methods:(a) H-Grid uses a similar method described in [9] where all valid hierarchical grids with more than 50 keypoints associated with the *Best Training Images* (identified using our MaxUnion-SKI method) are used for matching, (b) Top100 uses the Top 100 salient keypoints extracted using our MaxUnion-SKI method from the *Best Training Images*. (c) Top 200 uses the Top 200 salient keypints extracted using our MaxUnion-SKI method from the *Best Training Images*. For all three methods, we use the grid-based matching procedure described in [9] to process a query image.

TABLE II: Hierarchical Grid Result

	ACCY	Time (s)	P-Mem	V-Mem
H-Grid	91.3%	8.25	13GB	23.4GB
Top100	92.2%	0.26	389MB	1.14GB
Top200	93.0%	0.36	706MB	1.45GB

Table II shows our results. ACCY denotes the matching accuracy. Time is the matching time per query image. P-Mem is the allocated physical memory, while V-Mem is the virtual memory used. "H-Grid" achieves good matching accuracy but a single image needs more than 8 seconds for matching. Our experiment shows that the 232 *Best Training Images* generates 3410 hierarchical grids with around 1.6M keypoints/descriptors. Therefore, "H-Grid" consumes too much physical and virtual memory and slows down the matching significantly. "Top100" and "Top200" each achieves higher than 92% matching accuracy. However, the matching time (260ms for 100 keypoints) is slower than our MaxUnion-SKI method (95ms for 100 keypoints). The reason is that, for "Top100", 59% of the queries require full image matching before their landmarks are recognized but they incur wasteful processing cost for matching against smaller grids e.g. three 1x1, one 2x2 and one 3x3 grids without any success.

C. Server Throughput

We built two types of HTTP servers: a Synchronous and Asynchronous server. A Synchronous server handles one client request at a time, i.e. it must receive complete data from a

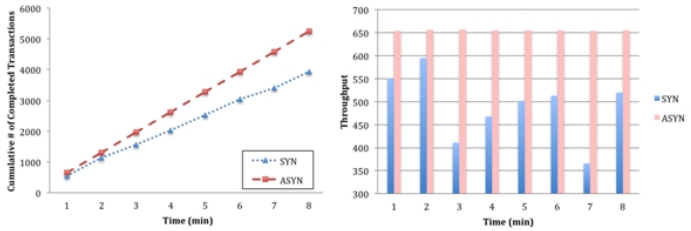

Fig. 9: Server Throughput

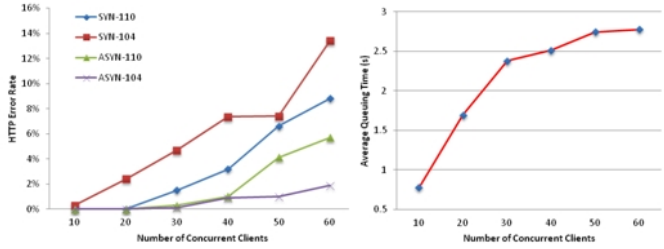

Fig. 10: Error Rates & Average Queuing Time

client before serving the next request. Meanwhile, the Asynchronous Server can handle multiple client requests without blocking. Since the synchronous server would be blocked until it has received a complete image, it can not handle too many concurrent client requests.

We did an experiment to evaluate the throughput of each type of servers. For this set of experiments, we only use the MaxUnion-SKI method with 100 selected salient keypoints. We simulate multiple clients sending image queries simultaneously from a laptop with a WiFi connection. Each client sends 100 image requests continuously, i.e. a client sends the next request after receiving the image matching result of the previous query image.

Figure 9 shows the throughput of the two types of servers. Here,we define throughput as the number of matching results returned per minute to the clients. With an Asynchronous server, it can achieve a stable throughput of 655 per minute. However, with a Synchronous server, the achievable throughput is quite unstable and may reduce to as low as 365 per minute. With a Synchronous server, the server cannot keep up with many client requests. Hence, some TCP connections ultimately time out and the number of completed requests drops.

Two types of HTTP errors occur during the process: Errno 110 (Connection Timed Out) and Errno 104 (Connection reset by peer). Errno 104 is a fatal error as the server side sends a RST packet to force close a connection. The left-hand plot in Figure 10 shows the error rates (percentage of errors for all client requests) of both Synchronous and Asynchronous servers. As seen in 10, though the error rates for both servers increase as the number of concurrent clients increases, the Asynchronous server does not incur any errors until 40 or more clients send requests concurrently. However, for a Synchronous Server, errors occur when there are only 20 clients in the system. With 60 concurrent clients, the error rate for a Asynchronous server is only 7.57% (including both Errno 110 and Errno 104) while the Synchronous server has an error rate of 22.22%. Meanwhile, many errors at the Synchronous server are the fatal connection reset errors which means that the server is forced to give up on many client requests.

When multiple clients send requests simultaneously, each request is inserted into a queue to be processed (image matching) by the server. Therefore, with increasing number of client requests, each request will incur additional queuing delay compared to the case when a client request arrives to see no queue ahead of itself. We conduct an experiment to measure the average queuing time when 100 requests are submitted per client to an Asynchronous Server. We use the 1st 20 requests as the warm up period, and only compute the average queuing time for the 21st to the 80th requests. As seen from the right hand plot in Fig10, the average queuing time initially increases rapidly as the number of clients increases but the increase rate slows down when the number of concurrent clients exceeds 30 because the server begins to drop some connections and hence the queue size does not increase as fast.

D. Query Image Preprocessing

We evaluate the impact of the image preprocessing scheme on the performance of our prototype system. For this experiment, we use the LeBuD image sets. 54 photos taken from various smarpthones are used as query images. Their sizes can be: 2560x1920, 1200x900, 1280x720. Some query images are large while some have "Green" content. The query images are resized to different scales: 240, 320, 480, 640. Each scale value determines the length of an image, and its width is scaled to maintain the original aspect ratio. Table III presents the results for the image preprocessing process. In the table, *"G-ACCY"* and *"NG-ACCY"* represent the matching accuracy for the "Green Cropped" and Non-cropped query images. The *"#Greened"* denotes the number of query images being cropped by the "Green Cropping" algorithm. The *"MatchingTime"* is the average server query matching time.

TABLE III: Client Preprocessing Effect

Scale	G-ACCY	NG-ACCY	#Greened	MatchingTime
640	100%	100%	16	72ms
480	98.15%	96.30%	16	47ms
320	96.30%	94.44%	17	26ms
240	90.74%	87.04%	17	20ms

From the table, we see that the matching accuracy increases when the "Green Scenery" is cropped from the query images. In addition, a larger image size of a good query image (from the right angle etc) usually means a higher matching accuracy but longer server processing time. With LeBuD, when the query image is resized to a scale of 320 (which is 50% of the training image size), we can still achieve a matching accuracy above 92%. However, when the query image is resized to a scale of 240 (which is 37.5% of the training image size), the matching accuracy drops to around 90%. For a large query image, randomly select 100 pixels to determine if it has green contents may not be sufficient. We intend to explore an appropriate setting in the near future.

E. Mobile Device Energy Cost

We evaluate the power consumption of our Android-based EMOVIS application running on a Samsung Google phone using *PowerTutor*, a per-application energy statistics application developed by University of Michigan. Our Android client is connected to a wireless router in our laboratory which is connected to the Lehigh University Campus Network. The measured WiFi signal using Network Signal Info tool was -53dBm with 72Mbps bandwidth. We evaluated the energy cost using two query images which are resized to two different sizes: 240*320 and 480*640. For each image size, we recorded the energy consumption of the Android client while

it preprocessed, and sent a query using that image 100 times continuously. We repeated such an experiment 5 times using each image. Our results are summarized in Table IV. Each reported value is the average value of these 5 trials. The average per image energy cost is about 0.4J if a query image is cropped to 480x640 size. This is a reasonable energy cost since streaming a short movie trailer of 1 minute duration can consume 68J on the same type of smartphones.

Different images of the same size have different energy cost depending on the extent of the "Green" content in an image, the original image size as well as the number of keypoints detected by our modified SURF-based algorithm. The more keypoints identified from an image, the longer time the server needs to do the matching, resulting in longer TCP connection times. Reducing the image size can significantly reduce the energy cost though we have to make sure that the cropped image is of sufficient quality to produce accurate matching.

TABLE IV: Energy Cost Per 100 Image Query (J)

	Size	CPU	WiFi	CPU+WiFi
img1	480*640	10.9	29.5	40.4
img2	480*640	11	39.3	50.3
img1	240*320	5.9	15.4	21.3
img2	240*320	9.3	17.6	26.9

F. Discussion

Here, we discuss a few techniques that can be used to further improve the performance of EMOVIS. One technique is to use location information (GPS coordinates) associated with the query image to reduce the number of salient keypoint sets that the EMOVIS server needs to compare with during the matching process. For example, a location-aware enhancement can find training images that lie within a certain region of that query location and only match those qualified images rather than the whole collection of training images. Furthermore, additional servers can be deployed to deal with increasing query loads for a large scale system. For a large scale system with many images, one can also deploy various servers to deal with different subsets of images, and conduct searches in parallel.For example, the Distributed Kd-trees[8] is a method which employs MapReduce architecture to efficiently build and distribute kd-trees to deal with millions of images.

In addition, optimization techniques suggested by other researchers can also be incorporated in our system. For example,an optimization method described in [18] which removes keypoints in a query image that match some training images from several landmark objects. Such an optimization method will improve the matching accuracy.

Furthermore, keypoints-based image matching has some inherent limitations. For example, different sets of keypoints can be detected for images from different angles of a layered building (e.g. a building with pillars in front) or a glass-structure building. Moreover, SURF-based keypoints are detected from gray-scale images, and thus it is insensitive to colors. Therefore, we can investigate how to combine keypoints features with color/texture features (e.g. CEDD [21]) to improve the matching accuracy.

VI. Conclusions and Future Work

In this paper, we have presented EMOVIS, an efficient and robust mobile visual search system, which we have designed. EMOVIS uses two salient keypoint identification schemes we designed to identify top 100 keypoint descriptors which can be used to uniquely identify a landmark, and hence improve matching accuracy. Furthermore, EMOVIS uses an image cropping algorithm to remove irrelevant regions from a query image, and hence reduces the query latency time and energy cost of using EMOVIS on mobile devices. Extensive evaluations using images from the popular Zurich database reveal that EMOVS achieves high accuracy ($> 92\%$) with low computational and energy cost. Our designed system provides higher accuracy and faster matching response time than two existing approaches. In the near future, we intend to implement three enhancements which we have identified that can further improve the performance of EMOVIS. Furthermore, we intend to evaluate our system using a larger image database e.g. the Manhattan database described in [2].

References

[1] G. Schindler, M. Brown, "City-scale location recognition", IEEE CVPR, 2007

[2] F. X. Yu, R. Ji, S.F. Chang, "Active Query Sensing for Mobile Location Search", ACM Multimedia, Nov-Dec, 2011.

[3] J. Knopp, J. Sivic, T. Pajdha, "Avoiding confusing features in place recognition",10th European Conference on Computer Vision, 2010.

[4] G. Baatz, K. Koser, D. Chen, R. Grzeszcuk, M. Pollefeys, "Handling urban location recognition as a 2d homothetic problem", 10th European Conference on Computer Vision, 2010.

[5] D. Chen, G. Baatz, K. Koser, S. Tsai, R. Vedantham,T. Pylvanainen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. "City-scale landmark identification on mobile devices", CVPR, 2011

[6] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, D. Schmalstieg, "Pose tracking from natural features on mobile phones", in ISMAR, 2008.

[7] N. Henze, T. Schinke, S. Boll, "What is that? Object Recognition from natural features on a mobile phone", Proceedings of Workshop on Mobile Interaction with the Real World, 2009.

[8] M. Aly, M. Munich, and P. Perona, "Distributed Kd-Trees for Retrieval from Very Large Image Collections", in BMVC 2011.

[9] W. Guan, S. You and U. Neumann, "Efficient Matching and Mobile Augmented Reality", ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP), speical issue on 3D Mobile Multimedia, Volume 8 Issue 3s, Article No. 47, September 2012.

[10] G. Fritz, C. Seifert and L. Paletta, "A Mobile Vision System for Urban Detection with Informative Local Descriptors", Proceedings of the Fourth IEEE International Conference on Computer Vision Systems, p.30, January 04-07, 2006.

[11] R. Boris, K. Effrosyni and D. Marcin, "Mobile museum guide based on fast SIFT recognition", 6th International Workshop on Adaptive Multimedia Retrieval, pp. 26–27, 2008.

[12] V. Pimenov, "Fast image matching with visual attention and surf descriptors," in Proceedings of the 19th International Conference on Computer Graphics and Vision, pp. 49-56, 2009.

[13] L. V. G. H. Shao, T. Svoboda, "Zubud-Zrich buildings database for image based recognition." ETH Zrich, Tech. Rep. 260, 2003.

[14] X. Chen , M. Koskela, "Mobile visual search from dynamic image databases", Proceedings of the 17th Scandinavian conference on Image analysis, Ystad, Sweden, May 01, 2011.

[15] J. J. Foo , R. Sinha, "Pruning SIFT for scalable near-duplicate image matching", Proceedings of the eighteenth conference on Australasian database, p.63-71, Ballarat, Victoria, Australia , January 30-February 02, 2007.

[16] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod. "Outdoors augmented reality on mobile phone using loxel-based visual feature organization", In Multimedia Information Retrieval, pages 427-434, 2008.

[17] Sergieh, H.M, Egyed-Zsigmond, E., Doller, M., Coquil, D., Pinon, J.-M. and Kosch, H., "Improving SURF Image Matching Using Supervised Learning", in SITIS 2012.

[18] X. Pan and S. Lyu, "Detecting image region duplication using SIFT features", Proc. ICASSP, 2010.

[19] Martin A. Fischler , Robert C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", Readings in computer vision: issues, problems, principles, and paradigms, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1987

[20] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in Computer Vision - ECCV 2006.

[21] S. A. Chatzichristofis, Y. S. Boutalis, "CEDD: Color and Edge Directivity Descriptor: A Compact Descriptor for Image Indexing and Retrieval" in ICVS 2008.