Mobile Object Detection through Client-Server based Vote Transfer

Shyam Sunder Kumar Min Sun Silvio Savarese
Dept. of Electrical and Computer Engineering, University of Michigan at Ann Arbor, USA
{shyamsk,sunmin,silvio}@umich.edu

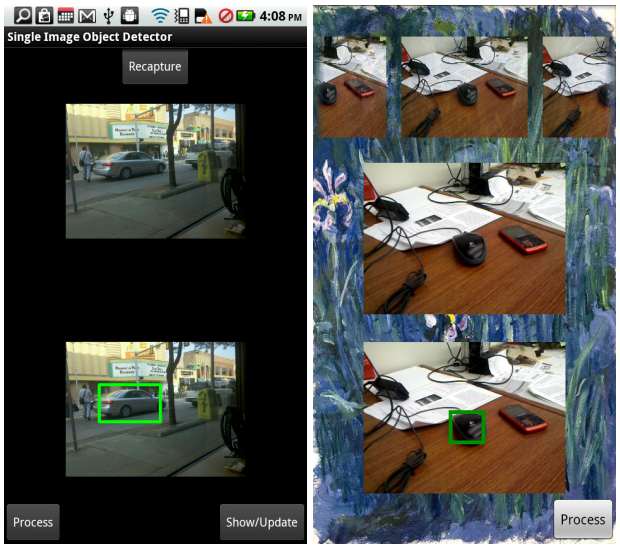
Abstract

Mobile platforms such as smart-phones and tablet computers have attained the technological capacity to perform tasks beyond their intended purposes. The steady increase of processing power has enticed researchers to attempt increasingly challenging tasks on mobile devices with appropriate modifications over their stationary counterparts. In this work we present a novel multi-frame object detection application for the mobile platform that is capable of object localization. Our work leverages the hough forest based object detector introduced by Gall et al. in [10]. In our experiments, we demonstrate that our novel, multi-frame generalization of [10] notably improves the detection performance. We test the performance of the technique in variable resolutions, the applicability to several object categories and different datasets. We implement the multi-frame detector on a mobile platform through a novel client-server framework that presents a sound and viable environment for the multi-frame detector. Finally, we study implementations of both single and multi-frame object detectors based on this client-server framework on a mobile device running the android OS.

1. Introduction

In the past ten years, mobile phones have steadily outmoded several conventional tools and devices by performing increasingly varied and complex tasks. Today it is possible to use a mobile device from finding a route to a destination, reading a book, browsing the web, to organising one's day. In recent years vision based tools such as barcode scanners and landmark recognition systems have made a positive impact on extending usability. Furthermore, single instance recognition systems have been used for tasks such as art (painting) recognition[22, 2], book cover and CD album cover recognition[11]. These applications extract simple features on-device and use powerful hashing strategies on a remote server to recover results. But such methods cannot be generalized for generic object category detection.

The ability to perform object detection on a mobile platform opens the door to compelling applications such as visual searching, object specific augmented reality, cataloging objects in a scene, object category specific recommendation systems, etc. Despite the range of opportunities, implementing an efficient and accurate algorithm for object detection on a mobile phone is an open challenge. Existing state-of-the art object recognition methods such as [7, 10, 6, 8, 13, 14] have been successfully applied to object detection and have been shown to obtain reasonable recognition rates. However, in practice, implementing and transferring such methods into a mobile platform is far from being an easy task. Mobile devices have limited memory and computational power, whereas methods such as [7, 10] have both significant computational and memory needs. Adapting these methods for a mobile phone therefore needs an effective split between the mobile phone (client) and a server. Moreover conventional object recognition methods are not designed to detect the object from multiple view-points. This ability is critical when:



(a) Single Frame (b) Multi Frame

Figure 1. Screenshots of our implementation of our proposed object detection techniques on the Android OS

1. The object is three dimensional - unlike a painting, or book cover, 3D objects vary considerably from differ-

ent view-points and often cannot even be viewed completely from a single vantage point.

2. Active Sensing scenarios are considered. In such scenarios the device can provide feedback to the user to point the camera from the most informative view point.

In our work we present a technique for leveraging the multiple frames towards improving object detection and further applying it in a mobile platform assuming multiple frames are extracted from a short video sequence. We define a *short video sequence* to consist of physically adjacent frames where the object remains in the field of view of the camera and smooth changes in the view of the object are introduced(Figure 2). Notice that, while steady motion is preferred, it is not instrumental for the technique to provide improvements.

The multi-frame detection approach we present is a generalization of the Class Specific Hough Forests framework for Object Detection proposed by Gall et al. [10] and the Generalized Hough Transform [3] to allow patches in a single image to vote for possible locations of the object in the image. We extend this by generating votes from different patches across multiple frames in a single frame through a technique we introduce as *vote transfer*. In this technique we associate patches across frames using tracking to link the voting spaces of the separate frames. We experimentally demonstrate that our multi-frame technique performs notably better than the single-frame version.

We also implemented the multi-frame detector on a mobile platform to demonstrate our multi-frame detector in its natural used case. In this contribution, we present a division of labor that requires a significant, non-trivial contribution from the client-side and heavy processing from the server (back-end). Through a number of timing analysis, we present the near-practicality of such a system. In section 5 we discuss this division of labor and in section 6 we present the implementation details.

In summary, our main contributions are:

- A novel hough forest based multi-frame object detection framework.
- Vote Transfer - a novel technique to integrate the detection process across multiple frames of a short video sequence through tracking.
- A client-server framework with a non-trivial client to perform object detection on the mobile phone which is significantly more than a simple image/video capture task.
- A comparative time performance study of three versions of the object detector - on a mobile device, on a desktop machine, the proposed client-server framework.
- An analysis of the effects of lowering image resolutions on detection performance and resulting system speed-up.

We next present work pertinent to the vote transfer technique in section 2. In section 2 we also discuss prior works implementing object detectors on mobile devices as well as other works that have explored client-server models. Section 3 introduces the single frame detector that our work builds on. Section 4 begins with the generalization of the single frame approach and discusses the vote transfer technique in detail. Section 5 provides details of the mobile application organization. Finally we provide experimental analysis of both the vote-transfer technique and the mobile application in section 6.

2. Related Work

Extending the Hough Transform to work with arbitrary shapes was first introduced by D.H. Ballard in 1981 [3]. Since then it has been successfully applied to the problem of single instance object detection [14] as well as object category detection [13]. The Implicit Shape Model (ISM) introduced by Leibe et al. [13] forms the basis for many part based object detection algorithms. The ISM learns a model of the spatial distribution of local patches with respect to the object center. During testing the learned model is used to cast a probabilistic vote for the location of the object center. ISM builds a codebook of patches by clustering patches based on appearance. Several modifications to the ISM have been proposed over the years. Maji et al. [16] propose a method that computes weights for the ISM using a max-margin technique. To overcome the computational drawbacks of building a codebook using clustering, Gall et al.[10] propose to learn a discriminative codebook using a random forest that they call a Class-Specific Hough Forest. Sun et al. [23] introduce the idea of Depth Encoded Hough Voting, that incorporates depth information in training in order to learn a one-to-one mapping between scene depth and patch scale. In our work, we do not introduce any modifications to the training set-up introduced by [10].

Thomas et al. [24] extend the ISM for multi-view object recognition, by learning many single view codebooks that they interconnect via what they call activation links. These activation links are obtained through the image exploration algorithm proposed by Ferrari et al. [9] for recognizing specific objects and establishing dense multi-view correspondence across multiple views. In some sense, this is similar to the idea we pursue in our multi-frame detector. However, while they build associations between codewords across multiple codebooks corresponding to multiple views, we build association between patches observed across multiple frames using tracking in test-time. Other works, such as [18, 21], present techniques that leverage videos, however these techniques are limited to single instance object

Figure 2. Short-Video-Sequence: This image-strip, an excerpt from the Car Show Dataset[19], demonstrates the definition of a short video sequence. Note the presence of the object in all frames of the sequence, and the smooth view-point changes across frames.

recognition. Similar to us, the work by [17] also leverages short video sequences. However, [17] focuses on estimating poses of object categories.

On the mobile front the spread of works dealing with object detection is limited. Hartl et al. [12] introduce an on-device segmentation and detection technique for 2D objects such as coins, keys, and screws. Belhumeur et al. demonstrate an object recognition system for identifying herbaria [4]. They implement their system on a head mounted display system. Also, both [12, 4] specialize in detecting 2D object categories however, our work focuses on detecting objects whose intrinsic shape is 3D.

[11] present a client-server approach to combined object recognition and tracking where they perform server-side recognition and recognition aided client-side tracking. [20] present an on-device technique that can recognize multiple objects based on tracking of objects based in videos. However both [20, 11] perform single instance object recognition; our work focuses on category level detection.

3. Single-Frame Detection

In this section we will briefly review the concept of Hough Forests as introduced by [10] to localize the object.

3.1. Hough Forest

The Hough Forest technique introduced by [10] is used to learn a direct mapping between the appearance of an image patch and its hough-vote. This is accomplished efficiently by using a random forest classifier. During training, each tree in the Hough Forest is constructed based on a set of patches $\mathcal{P} = \{\mathcal{I},\ c,\ d\}$, where $\mathcal{I}$ is the appearance of a patch, c is its class label and d is the distance between the patch center and the object center. The class label c is either 0 or 1 depending on whether the patch is a background or foreground patch. Each leaf node in the Hough Forest contains a collection of patches with similar characteristics that were observed during training, along with their offset vectors $D = \{d_i\}$ and class labels $C = \{c_i\}$.

Consider a patch $\mathcal{P}(\mathbf{y}) = (\mathcal{I}(\mathbf{y}),\ c(\mathbf{y}),\ d(\mathbf{y}))$ centered at a location $\mathbf{y}$ in the test image. After passing it through the random forest classifier, it is mapped to a leaf node that has a probability $p(c(\mathbf{y}) = 1|\mathcal{I}(\mathbf{y})) = \frac{\sum_{i=1}^{N} \delta(c_i=1)}{N}$ (corresponding to the foreground) and a voting direction $p(d(\mathbf{y})|c(\mathbf{y}) = 1, \mathcal{I}(\mathbf{y})) \propto \sum_{i=1}^{N} (\delta(d = d_i).\delta(c_i = 1))$, (wherein N is the total number of patches mapped to that leaf node) which will exclude background patches. In other words, patch $\mathcal{P}(\mathbf{y})$ votes for an object of class $c(\mathbf{y})$ at each location $\mathbf{y} - d(\mathbf{y})$ with probability $p(d(\mathbf{y})|c(\mathbf{y}) = 1, \mathcal{I}(\mathbf{y})).p(c(\mathbf{y}) = 1|\mathcal{I}(\mathbf{y}))$.

Let us consider the event $E(\mathbf{x})$ that the object lies at location $\mathbf{x}$ in the image. We are interested in computing the probability that the appearance of a patch $\mathcal{I}(\mathbf{y})$ yields the event $E(\mathbf{x})$ at a given position $\mathbf{x}$ in the image, i.e $p(E(\mathbf{x})|\,\mathcal{I}(\mathbf{y}))$. This probability can be expressed as,

$$p(E(\mathbf{x})|\mathcal{I}(\mathbf{y})) = p(E(\mathbf{x}), c(\mathbf{y}) = 1|\mathcal{I}(\mathbf{y})), \tag{1}$$

since patch $\mathbf{y}$ has to be a foreground patch (i.e $c(\mathbf{y}) = 1$) in order to cast a vote for the object. We can further expand the term on the right hand side of equation 1 to obtain

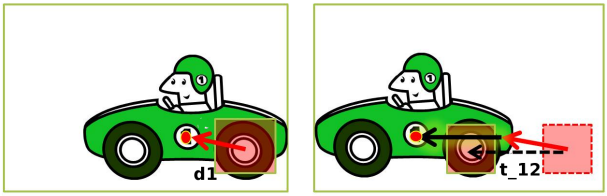
$$p(E(\mathbf{x})|\mathcal{I}(\mathbf{y})) = \tag{2}$$
$$p(E(\mathbf{x})|c(\mathbf{y}) = 1, \mathcal{I}(\mathbf{y})).p(c(\mathbf{y}) = 1|\mathcal{I}(\mathbf{y})) =$$

$$p(d(\mathbf{y}) = \mathbf{y} - \mathbf{x}|c(\mathbf{y}) = 1, \mathcal{I}(\mathbf{y}).p(c(\mathbf{y}) = 1|\mathcal{I}(\mathbf{y})))$$

Both terms on the right hand side of equation 2 can be obtained from the leaf node of the random forest into which the patch $\mathcal{P}(\mathbf{y})$ falls. The first term gives the probabilistic hough vote of the object center. This is obtained by a Gaussian Parzen density estimator to estimate the voting distribution (or alternatively a Mean Shift estimate) based on the offset vectors $D = \{d_i\}$ collected in the leaf during training. The second term is obtained from the proportion of object patches collected in the leaf during training (as opposed to background patches). The votes from different patches coming from the different random trees are added up to get a final score that can be visualized in terms of a heat map for the object location. More details are given in [10].

4. Multi-Frame Detection

The main contributing factor for considering the use of multiple frames for object categorization and detection is the potential presence of extra evidence that is not present in a single frame. The ready availability of such frames on mobile platforms makes this extra evidence more compelling to consider. However, the best approach to use this potentially advantageous evidence is not immediately apparent. In this section, we define multi-frame object detection problem and then we explore an approach that leverages this definition.



Frame 1 Frame 2

Figure 3. This figure illustrates the concept of transferring votes across frames. (**left**) Frame 1: A patch centered around the wheel of the car casts a vote for the object center in Frame 1 in a direction d_1 indicated by the red arrow. The patch center moves a distance t_{12} (indicated by the broken black arrow in Frame 2) between Frame 1 and Frame 2 to appear in the new position shown in the left panel. (**right**) Frame 2: The box with the dotted red outline shows the position of the patch in Frame 1, and the box with the solid yellow boundary shows its new position in Frame 2. In order for the patch in Frame 1 to cast a vote for the object in Frame 2, we displace its voting direction d_1 by the amount t_{12} (indicated by the solid black arrow)

4.1. The Multi-frame Problem

Consider a short video sequence with F frames. Denote a patch in frame $\mathbf{i}$ centred at location $\mathbf{y}$ as $\mathbf{y_i}$. Let $\mathcal{Y} = \{\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_F}\}$, be the set of patches that captures the motion of patch $\mathbf{y}$ through the video sequence. Similar to the single frame case , the existence of an object at location $\mathbf{x}$ in some frame $\mathbf{i}$ can be quantified by the conditional probability

$$p(E(\mathbf{x_i})|\mathcal{I}(\mathcal{Y})) \tag{3}$$

Here $\mathcal{I}(\mathcal{Y})$ denotes the appearance information of some set of patches $\mathcal{Y}$ as defined above. This formulation forms the cornerstone of the vote transfer method.

4.2. Detection by Vote Transfer

Here detection is performed by the accumulation of votes for the object center from several frames in-to the reference frame $\mathbf{i}$. Equation 3 can be re-written as $\sum_{\mathbf{y_j} \epsilon \mathcal{Y}} p(E(\mathbf{x_i})|\mathcal{I}(\mathbf{y_j}))$, which is equivalent to

$$\sum_{\mathbf{y_j} \epsilon \mathcal{Y}} p(E(\mathbf{x_j} + \mathbf{t_{ji}}(\mathbf{x}))|\mathcal{I}(\mathbf{y_j})) \tag{4}$$

wherein $\mathbf{t_{ji}}(\mathbf{x})$ denotes the displacement of the object at location $\mathbf{x}$ from frame $\mathbf{i}$ to frame $\mathbf{j}$. This is a reasonable approximation since a) we are only interested in the centroid of the object and b) we assume that the frames are derived from a short video sequence.

Notice that, in practice, displacement $\mathbf{t_{ji}}(\mathbf{x})$, is not readily available. Given the assumption of short video sequences, $\mathbf{t_{ji}}(\mathbf{x})$ can be approximated as the displacement of the vote cast by patch $\mathbf{y}$ by its motion across frames $\mathbf{i}$ and $\mathbf{j}$ (denoted by $\mathbf{t_{ji}}(\mathbf{y})$). This results in a transfer of votes from all frames of the video sequence to the reference frame $\mathbf{i}$ represented by $\sum_{\mathbf{y_j} \epsilon \mathcal{Y}} p(E(\mathbf{x_j} + \mathbf{t_{ji}}(\mathbf{y}))|\mathcal{I}(\mathbf{y_j})) = p(E(\mathbf{x_i})|\mathcal{I}(\mathcal{Y}))$. In this way, votes are transferred from frames $\mathbf{j}$ to frame $\mathbf{i}$ to detect the object in frame $\mathbf{i}$.

Similar to the single frame case, we can expand the probability $p(E(\mathbf{x_i})|\,\mathcal{I}(\mathbf{y_j}))$ as

$$p(E(\mathbf{x_i})|\,\mathcal{I}(\mathbf{y_j})) = p(E(\mathbf{x_i}),\, c(\mathbf{y_j}) = 1|\,\mathcal{I}(\mathbf{y_j})), \tag{5}$$

since patch $\mathbf{y_j}$ in frame $\mathbf{j}$ must be a foreground patch (i.e $c(\mathbf{y_j}) = 1$) in order to cast a vote for the object. We can further expand the term on the right hand side of equation 5 to obtain $p(E(\mathbf{x_i})|\,\mathcal{I}(\mathbf{y_j})) =$

$$p(E(\mathbf{x_i})|\,c(\mathbf{y_j}) = 1,\, \mathcal{I}(\mathbf{y_j})).p(c(\mathbf{y_j}) = 1|\,\mathcal{I}(\mathbf{y_j})) \tag{6}$$

We can rewrite the terms on the right side of equation 6 as,

$$p(d(\mathbf{y_j}) + t_{\mathbf{ji}}(\mathbf{y}) = \mathbf{y_j} - \mathbf{x_i}|\, c(\mathbf{y_j}) = 1, \mathcal{I}(\mathbf{y_j}))p(c(\mathbf{y_j}) = 1|\mathcal{I}(\mathbf{y_j})) \tag{7}$$

Similar to the single frame case, the term $p(c(\mathbf{y_j}) = 1|\mathcal{I}(\mathbf{y_j}))$ in equation 7 is the probability that patch $\mathbf{y_j}$ is a foreground patch given that it has an appearance $\mathcal{I}(\mathbf{y_j})$, and it can be obtained from the codeword to which the patch is assigned. The first term in the right hand side of equation 6 is the probability that the object center $\mathbf{x}_i$ in frame i is at a distance $d((\mathbf{y_j}) + t_{\mathbf{ji}}(\mathbf{y}))$ away from the location of the patch $\mathbf{y}_j$ in frame j. Remember that $d(\mathbf{y}_j)$ is the voting direction of patch $\mathbf{y}_j$. $t_{ji}(\mathbf{y}) = \mathbf{y}_j - \mathbf{y}_i$ is the motion of the patch $\mathbf{y}$ between frame j and frame i, and is obtained via tracking. In order to calculate this term, we displace the voting direction $d(\mathbf{y}_j)$ of the patch $\mathbf{y}_j$ by the amount of translation of patch $\mathbf{y}$ between the two frames $t_{ji}(\mathbf{y})$. This concept is illustrated in Figure 3. The first term of equation 7 can then be approximated by the Gaussian Parzen density estimator to estimate based on the displaced votes, similar to the single frame case. More details on the Parzen density estimator for a single image can be found in [10].

5. Mobile Application Blue-print

In this section we will describe our design for implementing our object detection algorithm on the mobile device. In order for the application to run within a practical amount of time and within the memory limits of the mobile device, we propose splitting the task of detection between the client and the server.

The mobile device is first used to capture either a single image or a short video sequence. The image / frames are then processed to extract the features used for object detection. If a multi-frame detection is performed, then pixel tracking is also completed on devices. This information is packaged and transferred to the server-side over an HTTP connection. On receipt of the features and tracking information, the server runs the vote-transfer detector. The result of this process is sent back to the client and displayed

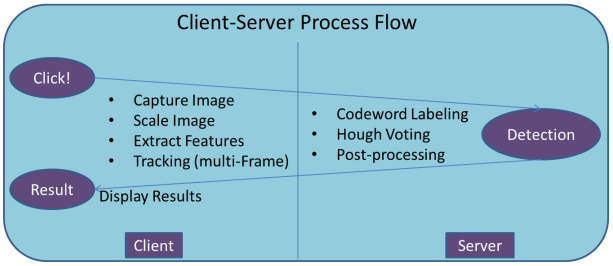


Figure 4. Client-Server Process Flow

to the user. The object detector runs detection over multiple categories in parallel. Data transmission between the client and server is conducted through a simple structured protocol. Figure 4 discusses the client-server process flow.

6. Experiments

In this section we present some qualitative and quantitative results of our multi-frame approach. We also provide timing analysis on both the mobile device (client) and desktop (server) to justify our proposed client-server model.

6.1. Datasets

We evaluate the performance of our algorithms to detect objects using two datasets. The first is a new multi-view dataset that we collected, and the second is the Car Show Dataset introduced by Ozuysal et al. in [19]. Please see the project web page (http://www.eecs.umich.edu/vision/MVproject.html) for more information about the dataset.

The new multi-view dataset contains videos of objects such as car, mouse, bicycle, and keyboard. There are 10 different instances of each object category, and the video sequences contain about 6000 frames per instance covering the entire viewing circle (360 degrees of the azimuth pose angle). Each video is sampled at approximately one image per 10 frames. Each frame contains a single instance of the object category. Additionally, the sampled images are approximately 0.6^0 (or 3^0 every 5 sampled frames) apart. The dataset also contains bounding box information for the sampled frames. For each object category, we train the Hough Forest using 420 images of positive and negative instances each including six instances, and use the remaining four instances for testing. Also note that the objects are observed under a cluttered background (Figures: 1,5). At no stage are images or object instances from testing used in the train set or vice-versa.

We also perform tests on the Car Show Dataset introduced by [19]. This dataset contains 360^0 images of 20 instances of cars with one image about every 3^0. The images are obtained as the cars are rotated on a turntable set-up. This dataset is particularly relevant due to the real world settings in which the images are recorded. Also, the larger frame-frame angular distance allows to further test the capability of our method. We use a 50-50 evaluation approach where we train the dictionary on ten instances with about 15 images per instance and the remaining ten instances are used for testing.

Finally, in all tests performed on both datasets, a positive detection is defined as 50% percent detection overlap with the ground-truth bounding box.

6.2. Vote Transfer

As discussed in section 4, vote transfer is the technique by which votes for the object centroid are transferred from subsequent frames in a video sequence to the reference frame. In this section we first discuss the implementation details of vote transfer and then we perform a series of comparisons that - a) demonstrate the improved performance of multi-frame detection, b) analyze tracking performance and, c) evaluate detection performance across various image scales.

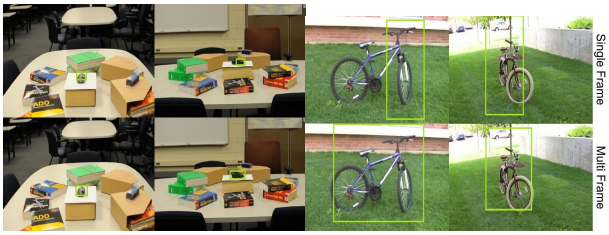
6.2.1 Implementation Details

The vote transfer mechanism introduced in Section 4.2 is tested against the single frame detection technique by [10]. During testing, we have a sequence of frames and we are interested in transferring votes from patches across this sequence of frames into a single frame that we call the reference frame. In our experiments we transferred votes from images that are 10, 20, 30, 40 and 50 frames away from the reference frame. These are the subsequent frames. After we accumulate all the votes in the reference frame, we add up the votes coming from all frames by giving a smaller weight to votes that come from frames that are farther away. Specifically, votes from patches in the reference frame receive a weight of 1, and votes from patches in the other frames receive a weight of $2^{-i/10}$ where $i = \{10, 20, 30, 40, 50\}$. Finally we identify the location of the object as the location with the maximum votes. These weights may also be learnt which we believe is a potential direction for future work.

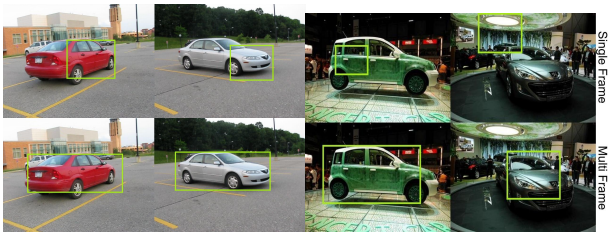
6.2.2 Single vs Multi-frame Performance

Now we quantitatively compare the performance of the single and multi-frame detectors. For the single-frame hough voting evaluation, we use votes from patches in the reference frame only. Figure 6 shows the precision-recall of the single and multi-frame detectors for the different object categories. Notice that for all object categories the multi-frame detector with vote transfer outperforms the single-frame detector.

Figure 5 shows a few qualitative examples of object detection for different object categories. The upper row shows the result of voting using patches in the reference frame alone, i.e single-frame hough voting. The bottom row shows results obtained by transferring votes from all frames in the sequence into the reference frame. Notice the

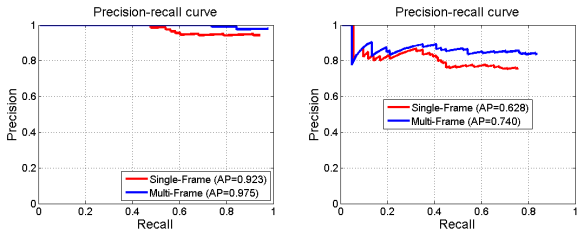


(a) Mouse on multi-view dataset (b) Bicycle on multi-view dataset

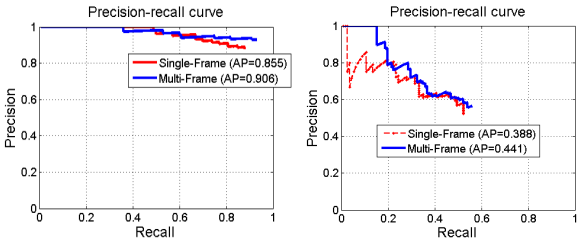


(c) Car on multi-view dataset (d) Car Show Dataset

Figure 5. For each instance the top row shows the result of single-frame hough voting on the reference frame. The bottom row shows results after vote transfer. Images that are 10, 20, 30, 40 and 50 frames away from the reference frame are considered.



(a) Mouse (b) Car



(c) Bicycle (d) Car Show Dataset

Figure 6. Precision-Recall curves for (a) mouse, (b) cars, (c) bicycles and (d) car - Carshow Data Set. Note that the multi-frame detector outperforms the single frame in all object categories.

multi-frame detector is able to localize the object better than the single-frame detector in many cases (Figure 5). This demonstrates that having more frames available for the voting process yields a positive affect on the detection process. Finally notice that the multi-frame technique performs well even for small objects (such as the mouse).

6.2.3 Tracking Analysis

Next we determine the effect of tracking by varying the separation between the reference frame and the subsequent frames. Moreover, we compare two tracking approaches a) Lucas-Kanade (LK) tracking (as implemented in OpenCV 2.1) [15] b) Large Displacement Optical Flow (LDOF) (as implemented by authors) [5]. It is critical to note that votes are transferred from the subsequent frames to the reference frame by computing the motion vectors from a subsequent frame to the reference frame. This prevents from accumulating errors during inter-frame tracking. Additionally, we do not use the votes from patches in the reference frame. This allows us to isolate the contribution of tracking from the rest.

This can also be understood as an evaluation of the detection performance in the reference frame by votes from a single subsequent frame j. Figure 7 shows a plot of the error as a function of the distance between the reference frame and increasingly distant subsequent frames. In red is the error curve when LK is used to track patches across frames, and in blue is the error curve for the case when LDOF is used to track patches. The error in the LDOF is small when the frames are close to each other, and it steadily increases as the distance between the frames grows. The error in LK increases gradually up to a point after which it remains steady. The sources of error in the votes transferred are as follows: 1) tracking is inherently noisy and this affects the way patches across frames are tracked. 2), the motion of a patch in the background of the image is unlikely to be the same as the motion of a patch on the object. Thus votes transferred from patches that belong to the background in the image inject noise into the detection process. Due to the relative stability of LK for larger inter-frame distances, we use this in our implementation. Also, the time taken to compute tracks (between two frames on a 2.4GHz triple core desktop computer) using LK is a matter of milli-seconds; the LDOF technique takes several seconds making it impractical.

Tracking using mobile based sensors: While mobile phones are rich in sensors such as gyroscopes, accelerometers and magnetic compasses, these sensors are not accurate enough to enable robust motion tracking. Theoretically, converting accelerometer signals to motion tracking is a double integral, but in practice, our tests demonstrated that the sensor readings introduced considerable drift. This led us to conclude that with the current state of these sensors, it is not suitable to perform motion tracking accurately.

6.2.4 Performance vs Image Resolution

Next we determine the optimal image resolution for detection. Since time and memory requirements for feature extraction increases with increase in the image resolution this is a key experiment to determine feasibility on a mobile platform. To do this, we run a performance analysis of the detector in both single and multi-frame versions on varying image resolutions. This experiment allows us to infer the optimal image resolution that provides a suitably high performance while minimizing the processing needs for feature extraction (see Figure 8). Our experiments show that while

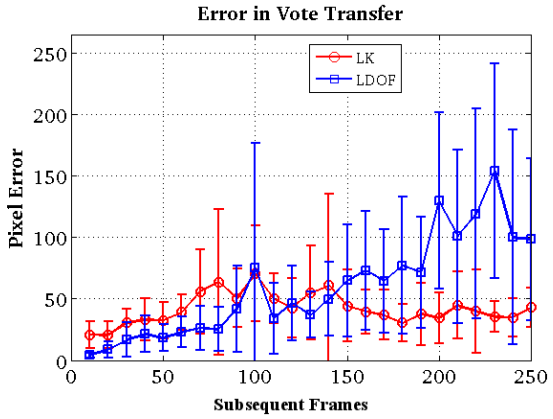


Figure 7. Tracking Error Plot vs Distance (with standard deviation): The error calculated is the distance between the ground truth bounding box center and the detected bounding box center. Here, the data point is the average error and the two end points show one standard deviation away from the average. We analyze LK (red curve) tracking vs LDOF (blue). LDOF has smaller error when the frame distance is small and increases steadily as expected. The error in LK is high to begin with, and increases gradually up to a point after which it almost stays steady. The error in the vote transfer plays an important role in the performance of our multi-frame detector.

best performance is achieved at a resolution of 640x480, the performance at 320x240 is comparatively very reasonable, whereas large deterioration in performance is observed at 160x120.

6.3. Mobile Platform

In this section we will first discuss some implementation details. This is followed by timing analyzes on the device only, desktop only, and the client-server model.

6.3.1 Implementation

We implement the object detector in three versions - a complete on-device single frame detector, a client-server single-frame application and lastly its multi-frame counter-part.

The complete on-device application waits for the user to capture an image. Once the image is captured, the image is processed for feature extraction through a pre-defined set of scales. Detection across multiple image scales leads to a scale invariant detector capable of identifying objects of different sizes. After this, the hough forest is loaded on-to the system memory and the patches are passed through the forest for label assignment. The patches then cast their votes for the object centre across all scales and possible aspect-ratios of the object. This is followed by post-processing involving non-maximum suppression of the voting space. Scales and aspect ratios to be explored are predefined based on object type. Also, the hough forests are trained and pre-loaded on-to the device. In this on-device implementation, we assume the object type is known and is capable of detecting cars, mouses, and bicycles.

In the remaining two versions, detection is run parallely for multiple objects. For both the client-server implementations, feature extraction across various scales is performed on device. Patch labeling, voting and post-processing are performed on the server-side. Additionally, for the multi-frame approach, tracking through LK tracking is performed on device as well. Finally, only extracted features from appropriate frames of the captured video (selected evenly across the video) are sent to the server side. This reduces the amount of bandwidth required to transfer information.

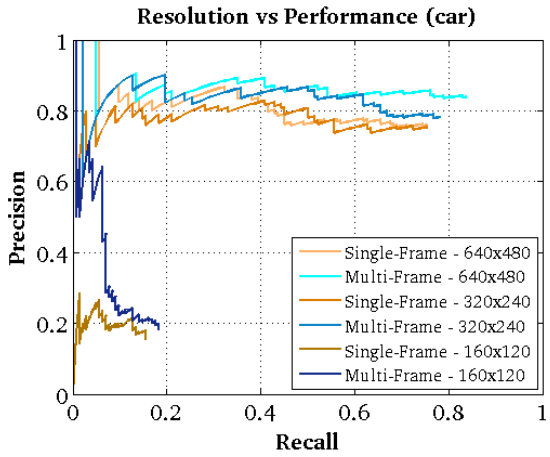


Figure 8. Image Resolution vs Detection Performance

6.3.2 Client-Server Analysis

In this section we will demonstrate the merit of a client-server system through some experiments. The tests where conducted on a Motorola Atrix (device specifications are available in [1]) running Android 2.2. on images of size 640x480 for detection. This was the best mobile configuration at the time of implementation. LK tracking is performed at the original image scale. Feature extraction recovers a 16-channel feature matrix and tracking consists of a 2-channel displacement matrix. We also test on a desktop machine with 2.4 GHZ triple core.

Tab.1 shows the times for traversing the random forest and conducting hough voting on both the mobile device and desktop machine. The detection process is significantly slower on device (∼3 to 4 times slower). The reason for the slower performance on the mobile platform is two fold: a) mobile devices have a slower, less powerful processor, and b) mobile devices lack a floating point core to perform floating point computations required for the detection step. We argure that this performance deficit justifies the use of a client-server model. In contrast to the mobile implementation, the client-server implementation (in Tab.2) significantly improves the performance from ∼6s (on device) to ∼3s for exploring a single scale on a single frame and traversing 10 trees.

Time (ms)	Random Forest	Hough Voting	Total
on device	19609	52666	∼70s
on desktop	6349	13872	∼20s

Table 1. Time break down of the detection process on a single frame exploring 8 scales and traversing 10 trees.

Time (ms)	LK	FE	CS	RF	HV	SC	Total
1 frame	N/A	300	650	456	1453	~20	~2.9s
5 frames	2430	1700	1200	6735	16773	~20	~28.9s

Table 2. Time break down of the detection process exploring 1 scale and traversing 10 trees. LK, FE, CS, RF, HV, and SC stand for Lucas-Kanade tracking, Feature Extraction, communication from Client to Server, traversing Random Forest, Hough Voting, communication from Server to Client, respectively.

7. Discussion

In this work we demonstrate a new approach to multi-frame object detection using Hough Forests and demonstrate through experiments the stability of this approach. The significance of this method is its ability to bring evidence from minor changes in views to improve object detection. To evaluate our technique, we introduce a new multi-view dataset that consists of 360^0 videos of three object categories. We further tested on the car-show dataset introduced in [19]. In addition to detection analysis, we also compare two tracking approaches and study accuracy by varying frame distances and also comment on time taken to compute. Through a study of performance on different image resolutions we further demonstrate the improved performance the multi-frame method provides over its single-frame counter-part. Finally, we demonstrate a realistic implementation of this technique on a mobile platform using a client-server approach.

There remains a rich scope for future work to add to our contribution. First, a more accurate and faster tracking technique will expand the applicability of this technique to beyond minor view changes. Secondly, this work has a natural extension leading to include pose estimation; however to use multiple views for this requires understanding how view-point changes can foster pose estimation. Also, compression of the feature and tracking matrices can aid reducing transmission time for the mobile implementation.

8. Acknowledgements

We acknowledge the support of a Google Research Award and the Gigascale Systems Research Center. We also would like to thank the contribution of Anush Mohan for his work on vote transfer and Giovanni Zhang for his work on the mobile implementation.

References

[1] Motorola atrix 4g - dual-core phone - android smartphone - tech specs - motorola mobility, inc. 7

[2] Google goggles. Web, 2009. 1

[3] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981. 2

[4] P. Belhumeur, D. Chen, S. Feiner, D. Jacobs, W. Kress, H. Ling, I. Lopez, R. Ramamoorthi, S. Sheorey, S. White, et al. Searching the world's herbaria: A system for visual identification of plant species. In *ECCV*, 2008. 3

[5] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *CVPR*, 2009. 6

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, Washington, DC, USA, 2005. IEEE Computer Society. 1

[7] P. F. Felzenszwalb, R. B. Girshick, and D. Mcallester. Cascade object detection with deformable part models. In *CVPR*, 2010. 1

[8] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*, 2005. 1

[9] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(1):36–51, 2008. 2

[10] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *CVPR*, 2009. 1, 2, 3, 4, 5

[11] S. Gammeter, A. Gassmann, L. Bossard, T. Quack, and L. Van Gool. Server-side object recognition and client-side object tracking for mobile augmented reality. In *CVPRW*, 2010. 1, 3

[12] A. Hartl, C. Arth, and D. Schmalstieg. Instant segmentation and feature extraction for recognition of simple objects on mobile phones. In *CVPRW*, 2010. 3

[13] B. Leibe, A. Leonardis, and B. Schiele. An implicit shape model for combined object categorization and segmentation. In *ECCVW*, 2004. 1, 2

[14] D. G. Lowe. Local feature view clustering for 3D object recognition. In *CVPR*, 2001. 1, 2

[15] B. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. *International joint conference on artificial intelligence*, 1981. 6

[16] S. Maji and J. Malik. Object detection using a max-margin hough tranform. In *CVPR*, 2009. 2

[17] L. Mei, J. Liu, A. Hero, and S. Savarese. Robust object pose estimation via statistical manifold modeling. In *ICCV*, 2011. 3

[18] N. Noceti, E. Delponte, and F. Odone. Spatio-temporal constraints for on-line 3D object recognition in videos. *Computer Vision and Image Understanding*, 2009. 2

[19] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *CVPR*, 2009. 3, 5, 8

[20] M. Raptis and S. Soatto. Tracklet descriptors for action modeling and video analysis. In *ECCV*, 2010. 3

[21] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Segmenting, modeling, and matching video clips containing multiple moving objects. *IEEE transactions on pattern analysis and machine intelligence*, 2007. 2

[22] B. Ruf, E. Kokiopoulou, and M. Detyniecki. Mobile museum guide based on fast SIFT recognition. *Adaptive Multimedia Retrieval. Identifying, Summarizing, and Recommending Image and Music*, 2010. 1

[23] M. Sun, G. Bradski, B. Xu, and S. Savarese. Depth-encoded hough voting for joint object detection and shape recovery. In *ECCV*, 2010. 2

[24] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiel, and L. Van Gool. Towards multi-view object class detection. In *CVPR*, 2006. 2