Object Detection on the Android
Anush Mohan, Shyam Kumar
November 4, 2010

1 Introduction:

Object detection has been a formidable challenge in computer vision. Although there has been considerable work that demonstrates good results when either the object was observed in training or when there is limited intra-class variability, a universal inter-class, intra-class solution remains unseen.

The case for further work to refine and improve object detection is transparent. Object Detection is a necessary step towards performing a myriad of other vision and machine automation tasks. In vision, applications include accurately understanding scenes, as a prior to interactions between objects/subjects (such as a person holding a microphone might be giving a speech), and others. In the broader problem of machine automation, object detection enables machines to understand and thereby interact (or not) with its environment.

Our object detection framework is largely based on the frameworks proposed in [1] and [4], which are both based on the idea of using the Generalized Hough Transform to allow patches in the image vote for possible locations of the object. In [4], the authors propose to use Random Forests to learn a discriminative codebook of part appearances that maps image patches to probabilistic votes about the object location. [1] extends this idea to incorporate scene depth into the framework by using depth maps in training, to build a mapping between patch scale and scene depth. This allows them to simultaneously detect objects and recover object shape. The advantages of deriving these two quantities at once are plenty. Depth information of a detected object aids as a base for applications in Augmented Reality, motion planning, scene understanding and others. However, our immediate goal is focused on object detection alone. The challenges involving incorporating depth extraction have been tackled for single frame queries in [1].

In this project, the focus is primarily on improving the performance of the joint object detection and pose estimation framework outlined in [1] using multiple frames. Towards this end, we propose the use of short video sequences (1-2s) of the object of interest. In addition to this, we will also implement a client-server framework between a mobile platform and a computationally more powerful fixed system.

2 Technical details:

Our goal is to improve the performance of the detector introduced in [4] by using a short video sequence instead of a single frame to make a detection hypothesis in a specified frame. Instead of using patches from a single frame to cast a probabilistic vote about the object location, we would like to use patches across frames to vote for the object location in a specific frame. In order to achieve this we need to develop a mechanism to transfer votes across frames. Before we discuss our solution for this problem, let’s first look at what it means for a patch to cast a vote in a single frame.

2.1 Patch voting

This section follows largely from Section 4 in [4]. Consider a patch $y$ in the query image that is matched to a codebook entry $I(y)$. Let $c(y)$ denote the class of patch $y$. For simplicity we assume that $c(y) = \{1,0\}$ where 1 indicates foreground and 0 indicates background. Let $E(x)$ be the event that the object center lies at location $x$. Then, the probabilistic vote for the object location cast by patch $y$ is:

$$p(E(x)|I(y)) = p(E(x), c(y) = 1|I(y))$$

(1)

since the existence of an object at $x$ means that $y$ was a foreground patch.

Then

$$p(E(x)|I(y)) = p(E(x)|c(y) = 1, I(y))p(c(y) = 1|I(y))$$

(2)

The first term on the right gives the probabilistic Hough vote of the object center. This is obtained by a Parzen Window estimate (or alternatively a Mean Shift estimate) based on the offset vectors $D_l$ 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). For more details see [4]. 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.
2.2 Transferring votes

Let’s say we want to use a patch in frame $j$ to cast a vote for and object position in frame $i$. Essentially, we’re after $p(E(x_i|I(y_j))$. If we can get these probabilities for different patches in different frames, then we can combine the votes across multiple frames to make a hypothesis in a specific frame. Since the voting is across frames, where each frame is a voting space, what we are after is a mechanism to transfer votes across frames or voting spaces. We propose the following strategies to achieve this.

2.2.1 Strategy 1

The basic idea is to make independent votes for the object center in each frame, and then transfer these votes to the given frame based on the motion of the object center position that we obtain using optical flow (see section 2.3). If our goal is to detect the object in frame $i$ then for all frames $\{j\}$ we hypothesise about the location of the object center in each frame $\{j\}$. We use optical flow to determine how the object center $x_j$ has moved between frame $\{j\}$ and $i$ in order to obtain $x_i$. More formally,

$$
P(E(x_i) | I(y_j)) = p(E(x_i), c(y_j) = 1 | I(y_j))$$

$$
P(E(x_i) | I(y_j)) = p(E(x_i) | c(y_j) = 1, I(y_j)) \cdot p(c(y_j) = 1 | I(y_j))$$

where, $t_{ij}(x) = x_i - x_j$ is the displacement of pixel $x$ (object center) between frame $i$ and frame $j$. This gives:

$$
P(E(x_i) | I(y_j)) = P(t_{ij}(x) = x_i - x_j) \cdot p(E(x_j) | c(y_j) = 1, I(y_j)) \cdot p(c(y_j) = 1 | I(y_j))$$

since the displacement of a pixel $x$ between two frames is independent of the fact that it is the object center. Further, since the displacement is also independent of the appearance of $y$ $I(y_j)$ or the fact that it is a foreground patch, we get

$$
P(E(x_i) | I(y_j)) = p(t_{ij}(x) = x_i - x_j) \cdot p(E(x_j) | c(y_j) = 1, I(y_j)) \cdot p(c(y_j) = 1 | I(y_j))$$

Here, the first term is the confidence of the tracking algorithm since it gives the probability that pixel $x$ was tracked from frame $i$ to frame $j$. The last two terms are the same as equation (1), i.e. it is obtained by running the Hough Forest detector[4] directly on frame $j$.

2.2.2 Strategy 2

Here we try and follow the intuition developed in [5] about linking patches observed across multiple views. However in our case links are established using optical flow, and they are established across multiple frames in testing (unlike [5] where the links are established in training). Following the steps developed in the previous section

$$
p(E(x_i) | I(y_j)) = p(E(x_i), c(y_j) = 1 | I(y_j))$$

$$
p(E(x_i) | c(y_j) = 1, I(y_j)) \cdot p(c(y_j) = 1 | I(y_j))$$

marginalize over all patches $y$ in frame $i$.

$$
\sum_{y_j} p(E(x_i), I(y_i) | c(y_j) = 1, I(y_j)) \cdot p(c(y_j) = 1 | I(y_j))$$

$$
\sum_{y_j} p(E(x_i) | c(y_j) = 1, I(y_j), I(y_i)) \cdot p(I(y_i) | c(y_j) = 1, I(y_j)) \cdot p(c(y_j) = 1 | I(y_j))$$

$$
\sum_{y_j} p(E(x_i) | c(y_j) = 1, c(y_j) = 1, I(y_j), I(y_i)) \cdot p(I(y_i), c(y_i) = 1 | c(y_j) = 1, I(y_j)) \cdot p(c(y_j) = 1 | I(y_j)) \cdot p(c(y_j) = 1 | I(y_j))$$

(patch $y$ in frame $i$ must be an object patch since it votes for the object at location $x_i$.

$p(c(y_j) = 1 | I(y_j))$ is the probability that $y_j$ is a foreground patch, and $p(c(y_i) = 1 | I(y_i))$ is the probability that $y_i$ is a foreground patch.

$p(I(y_j), c(y_j) = 1 | c(y_j) = 1, I(y_j))$ is the probability that a patch $y_j$ has an appearance $I(y_j)$ in frame $i$ and appearance $I(y_j)$ in frame $j$. This term can be learned in training. Alternatively we can model this probability as a Gaussian with a mean of $\gamma_j$ and variance proportional to the error in optical flow tracking (i.e. larger for frames that are far away). $\gamma_j$ is the location of $y_j$ back-projected in frame $i$ using the optical flow.

$p(E(x_i) | c(y_j) = 1, c(y_j) = 1, I(y_j), I(y_i))$ is the vote for object position $x_i$ in frame $i$ made by patches $y_j$ and $y_i$, which can be determined by treating the votes from $y_j$ and $y_i$ independently. This probability is non zero only for those patches $y_j$ which could have moved between frames $i$ and $j$ to become $y_j$. The voting direction for $y_j$ is displaced by $t_{ij}(y) = y_i - y_j$, which is the optical flow between $y_j$ and $y_i$. Once patches $y_j$ and $y_i$ have voted for the object position, Parzen Window estimation is used similar to [4] to determine the probability.
2.3 Patch Tracking

We use the algorithm developed in [3] to track pixels across frames.

2.4 Client-Server Framework

As mentioned in the introduction, in addition to the object detection framework, we will also implement a client-server framework between a mobile platform and a computationally more powerful fixed system. The main task of the client would be to acquire video sequences and perform limited pre-processing steps. This combination would then be sent over to the server to detect objects and their pose. A number of high confidence detections will be returned to the client for display to the user.

2.5 Challenges

There are a number of technical and engineering challenges associated with our proposed technique.

1. Firstly, tracking patches accurately across various frames as proposed in section 2.3, is a computationally expensive process. Even short video sequences can take exponentially long times (wrt video length). Using faster algorithms lead to results with low accuracy. In order to use noisy algorithms, it would be necessary to develop noise reduction strategies.

2. Secondly, the number of pre-processing steps that may be implemented on mobile devices is limited by the low memory and inherently misaligned architecture of off-the-shelf mobile platforms. This may limit us to using mobile devices as a sensor alone.

3 Milestones completed:

1. New multi-view dataset for training and testing obtained:

   Dataset contains instances of various objects, for example mouse, car, keyboard, etc. Each object category has multiple instances (6~10). The viewpoint changes for each object instance is smoothly distributed over around 4000 images, extracted from a one to two minute video. The images in this dataset also have annotations of object locations and pose. However this dataset currently has no depth information available.

2. Single frame detection up and running

   We have adapted the Depth Encoded Hough Voting implemented by [1] to work with this new dataset. Specifically, we have modified it to train without depth information. Currently we can recover the top five candidate detections across all poses for a given query image.

3. Pixel tracking up and running

   We have tested the Large Displacement Optical Flow method proposed by [2] to track pixels across frames in a video sequence. This is our preferred candidate tracking algorithm (ex: KL tracker implemented in OpenCV) as it is state-of-the-art.
4. Formulated new framework to transfer votes.

We have formulated a probabilistic framework to transfer votes from patches in one frame to a reference frame. See section 2 for details.

4 Remaining milestones

Our adapted DEHV code is capable of generating a hypothesis for an object category in a given query image. Our goal builds on this by utilising video sequences instead of single frames. We are in the stage where we have to build on top of the existing code:

1. Implementation of Vote Transfer framework:
   (a) Implement Option-1 (see Section 2.x) - [14 November 2010]
   (b) Implement Option-2 (see Section 2.x) - [24 November 2010]
   (c) Interface pixel tracking with hough-voting. - [14 November 2010]

2. Test Detection Framework:
   (a) Run a suitable batch of tests to analyse the robustness of our proposed technique [1 December 2010]

3. Create Client-Server framework:
   (a) Create client-side interface to acquire a video sequence (of a suitable length). [4 December 2010]
   (b) Create a server-side framework to receive input from client, perform object detection and report back top five candidates to client. [8 December 2010]

4. Report [17 December 2010]

5 References


2. A. Leibe, A. Leonardis, B Schiele, Combined object categorization and segmentation with an implicit shape model, Workshop on Statistical Learning, ECCV 2004


4. Gall J. and Lempitsky V., Class-Specific Hough Forests for Object Detection (PDF), CVPR 2009