Unsupervised Learning Segmentation of Objects in a Scene

Yi Tian Xu
260520039

April 18, 2017

Abstract

1 Introduction

Object manipulation and cognition are arguably a highly coupled task. For example, human infants can develop their manipulation skills through imitation [1]. For an autonomous robot, we thus image it to be driven by a continuous sequence of action, perception and reaction. In particular, it needs to identify and reason about its surrounding objects that potentially lie in a dynamic and unstructured environment.

Recent studies in neural networks, in particular, Fully Convolutional Network (FCN) [2] and Mask RNN [3], have risen as powerful tools in segmentation. Yet, as any supervised methods, a pre-defined labelled dataset provided by a human teacher may limit a robot in reasoning about novel objects. Moreover, robots who work in a specialized field, for example, pruning of city trees, would have supervised visual data that can be expensive to obtain, while robots with multi-dimensional specialities, for example a eldercare home robot, may require an extensive amount of data to cover all cases exhaustively.

In this work, we address to the problem of objects segmentation based on an unsupervised approach. We use motions to determine object boundaries and a probabilistic matching method to track the segments. In our setting, we will consider a passive robot that observes motions in its environment.

2 Related Works

2.1 Segmentation Through Action

Past studies have attempted to used robot's actions to coordinates its reasoning in detecting segments. For example, Hoof et al. [4] partitioned the environment into parts tracked by local key point descriptors, and developed a probabilistic model to infer the labelling of each part and to select best action to maximize information gain. As their approach rely on visual features detectable through Scale Invariant Feature Transform algorithm, the robustness to textureless objects is weak. Another disadvantage is that they ignore continuous observation of the objects and hand motion. Instead, they took snapshots of the scene before and after a robot action, and use Random Sample Consensus algorithm to find a rigid 3D transformation to track the parts. Due to a 1% inaccuracy in their tracking algorithm, the tracking error was accumulated action by action, and prevented the robot to correctly infer the number of segments in their experiment.

Hoof et al. mentioned that their method is independent of the type of descriptor employed provided that the set of key points is sparse and each can be detected reliably from multiple views. Thus, expanding their method to low-texture object may simply require adding a new type of descriptors. However, we suggest that the use of continuous motion observation can provide enough information to track the parts without tracking key points. The main concern is to segment the hand from the object.

Correlation between the hand movement and the moved object may impose a challenge in discriminating the robot hand from touched object. Fitzpatrick et al. [5] experimented on segmenting one object of interest by making an impact movement on that object. They suggested to isolate the motion of the arm from the motion of the targeted object by comparing the image differencing before and after the moment of impact. We adopt their idea of using image differencing and attempt to expand their method to general hand-object movements and co-movement of objects.

2.2 Gabor Filters

As image differencing relies on the difference between pixel values in consecutive frames, it allows us to obtain a thin object boundary depending on the direction and the speed of the motion. Fitzpatrick et al. [5] used a maximum-flow algorithm for vision [6] to find the interior region of the object boundary. Our method will try to obtain a thicker object boundary allowing us to quickly fill the interior of the object. To this end, we employ gabor filters [7] to magnify the edges prior the comparison between frames.

2.3 Segment Tracking and Matching

2.4 Image Mutual Information

Mutual Information (MI) as been a common measurement tool for image comparison in many clinical applications [8], mainly for aligning 2D images to form 3D images. At the heart of the technique is the approximation of the image distribution using the marginal and joint histogram. In this formulation, each pixel is a random variable. The marginal histogram counts the occurrence of pixel values in the image, while the joint histogram counts the co-occurrence of pixel pair values in two image, which is a 2D histogram.

Marginal and joint histograms are then used to compute marginal and joint entropy, which are subsequently used to compute MI. Maximizing MI over images transformation (e.g.: rotations and translations) approximates the best alignment. This assumes a statistical relationship between two images which can be captured by the joint entropy. Maximizing MI is thus related to minimizing the joint entropy. Yet, the joint entropy may encounter some edge case problem when used alone, as argued by Pluim et al. [8]. (?)

The independence of visual features and the lack of parameter tuning have given mutual information success in the medical domain. Pluim et al. showed in their survey a diversity of variation of MI, including normalized MI and MI that includes spacial information by consider one neighbouring pixel value. Russakoff et al. [9] proposed regional mutual information, extending the idea of spacial information inclusion by considering all neighbouring pixel valus in a nxn neighbourhood, creating $n \times n$-dimensional marginal histograms and $2n \times n$-dimensional joint histograms. Computational complexity can be optimized using PCA and the performance is asymptotically close to the original MI as the neighbourhood radius is fixed.

Although Russakoff et al. showed that RMI is more robust to noise than the original formulation of MI and the variations explored by Pluim et al, we will use the original formulation of MI as a measure to match segments between frames and to experiment its performance in our problem context.

3 Method

3.1 Filtering and Part Extraction

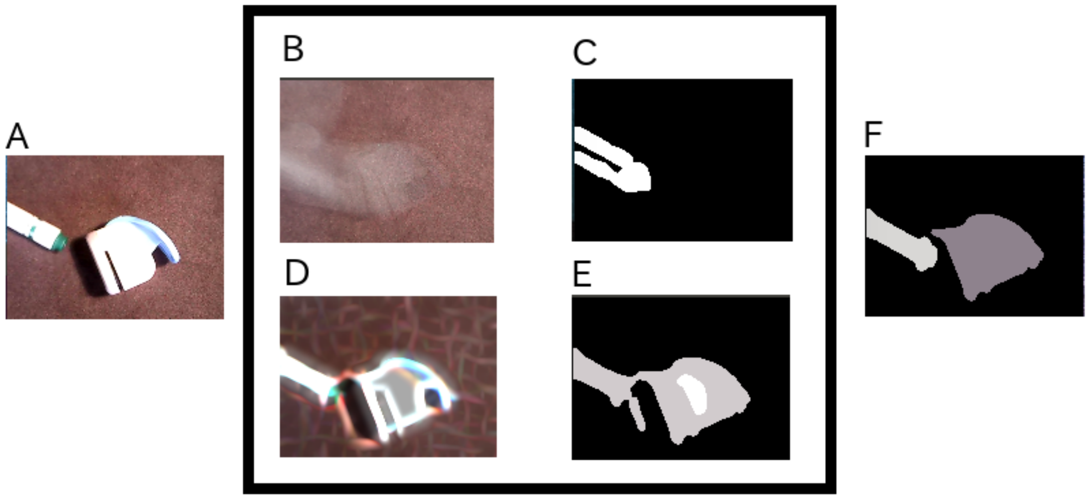
To allow the gabor filter running close to real-time, we had to downside the image to 160 $\times$ 120 px.

At each frame, we add it to the background average and we use image differencing to detect motion. When a considerably large motion is detected, we preform gabor filtering on the background average and the most recent frame. In our experiment, we used $n_g = 55$ differently oriented gabors of $s_g = 12$ px in size. Only cosine gabors are considered although integration of sine gabors is possible. (and inversed cosine gabors) A second image differencing between the two filtered images, which is equivalent to a background subtraction, yields isolation of moved edges.

To obtain a clean image of the moved edges, we had to add a few other filters before and after the gabor filters, namely Gaussian blur and thresholding. Below is the pseudocode for parts extraction.

```
function PartsExtraction(RGB image frame $I$, RGB background average $B$)
    $I' \leftarrow$ GaborFilter($I$)
    $B' \leftarrow$ GaborFilter($B$)
    $P \leftarrow$ AbsoluteDifference($I'$, $B'$)
    $P \leftarrow$ GaussianBlur($P$, $\sigma$)
    $P \leftarrow$ Thresholding($P$, $\theta_1$)
    $P \leftarrow$ SplitIntoContours($P$, MIN_COUNTOUR)
    return $P$
end function
```

Figure 1: Visualization of the method: (A) Input frame is being processed to (B, C, D and E). (B) Background average is accumulated with the new frame. (c) Image difference detects movement. (D) Gabor filtering detects edges in the current frame (also on the background (B), not shown in image). (E) Image differencing between gabor filtered background and (D) results image parts that can be mapped to segments. (F) Finally, a probabilistic matching method infer the labellings of the parts in (E) according to the previously inferred segments.



```
function GaborFilter(Image $I$)
    $I_1 \leftarrow$ GaussianBlur($I$, $\sigma$)
    $I_2 \leftarrow I_1$
    for each gabor kernel $k$ do
        $I_2 \leftarrow$ max(Convolution($I_1$, $k$), $I_2$)
    end for
    $I_2 \leftarrow$ Thresholding($I_2$, $\theta_2$)
    return $I_2$
end function
```

The Gaussian blur preforms a Gaussian averaging for each pixel with a neighbouring spread of σ. The thresholding segments the image into two parts with the cutoff pixel value θ_i. The parameter σ is tuned to 11 in the experiment. θ_1 and θ_2 are set to be the standard deviation of the pixel values for the gabor filtered frame image and the mean increased by t times of the standard deviation of the gabor filtered frame image respectively. t is set to 1.7 in the experiment.

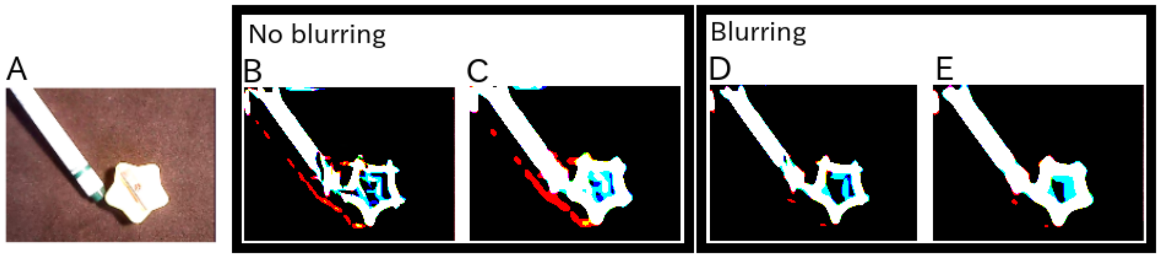
The size of the Gaussian blur (σ) affects the performance of the gabor filters. We added blurring prior to the gabor filter to soften the shadow and to smooth the extracted edges (see Figure 2). A additional Gaussian blur is used after the gabor filter accompanied with thresholding to smooth further the edges and to remove noise.

The last operation in the parts extraction function, *SplitIntoContours*, finds the contours and fill-in colors. This is a quick heuristic to segment moved object from the background, assuming that all objects with no visible holes (topology (?)...). The obtained segmentation is used a partitioning of the image that shall feed the probabilistic matching method to infer and track the final object segments.

3.2 Probabilistic Segment Matching

Both the motion detected through image differencing (which we will call motion image henceforth) and the image with the extracted parts are considered in the probabilistic matching model. Intuitively, t

Figure 2: Comparison between non-blurring(B, C) and blurring (D, E) prior to gabor filtering. (A) The original scene frame. (D) Detected edges without Gaussian blur prior to gabor filtering, and (C) with Gaussian blur and thresholding after gabor filtering. (D) Detected edges with Gaussian blur prior to gabor filtering, and (E) finalized with another Gaussian blur and thresholding.



3.2.1 One-to-One Matching

3.2.2 Merging Segments

3.2.3 Splitting Segments

4 Experiment

5 Discussion

Use binary search to first to see which region in a larger scene a moved object can be matched to.

Allow robot to grab object and observe the patterns and shapes which can be learn with a FCN. Allow communication between human teacher and a robot where teacher points and name the object and robot can go grab that object and learn it.

6 Conclusion

References

[1] A. N. Meltzoff *Infant Imitation After a 1-Week Delay: Long-Term Memory for Novel Acts and Multiple Stimuli.* Developmental Psychology (1988) Vol 24, No 4, 470-476.

[2] J. Long, E. Shelhamer, and T. Darrell. *Fully convolutionalnetworks for semantic segmentation.* CVPR, 2015

[3] K. He, G. Gkioxari, P. Dollàr, R. Girshick *Mask R-CNN.*

[4] H. V. Hoof, O. Kroemer, and J. Peters *Probabilistic Segmentation and Targeted Exploration of Objects in Cluttered Environment.*

[5] P. Fitzpatrick and G. Metta *Grounding Vision Through Experimental Manipulation.* Philos. T. Roy. Soc. A (2003) Vol. 361, No. 1811, 2165-2185

[6] Y. Boykov and V. Kolmogorov *An Experimental Comparison of Min-Cut/Max-Flow Algorithm for Energy Minimization in Vision.* Energy Minimization Methods in Computer Vision and Pattern Recognition (2001) 359-374

[7] R. Mehrotra *Gabor Filter-Based Edge Detection.*

[8] S. Park and J. K. Aggarwal *Segmentation and Tracking of Interacting Human Body Parts Under Occlusion and Shadowing.* Workshop on Motion and Video Computing, Orlando, Florida (2002)

[9] J. P. W. Pluim *Mutual Information based Registration of Medical Images: a Survey.*

[10] D. B. Russakoff *Image Similarity Using Mutual Information of Regions.*