Object Tracking Using Modified Color Indexing and Edge Feature

University of Minnesota
Meng Chen
April. 28th 2016

ABSTRACT

Using drone as the aerial platform and computer vision as the tool, we can accomplish many tasks autonomously, such as conducting powerline and pipeline inspection, surveillance, land survey, target tracking and search and rescue missions. This project focuses on the computer vision tools and techniques that are fundamental in achieving some of those tasks, particularly, object identification and tracking. The goal is to create program that can detect and track and an object using a given template. The features used includes color and edge density.

INTRODUCTION

As unmanned aerial vehicle (UAV) becomes more and more ubiquitous, it is also transforming how works are done in many fields. In agriculture, drones enable large area yield estimates and precision agriculture; in geology and mapping, the usage of drones offers higher precision for lower costs. In some search and rescue missions, UAV could vastly expand the search area while reducing the man power and resources required. That is not the exhaustive list, but it is worth noticing that most applications involve a UAV carrying a camera. In fact, UAV is becoming the universal platform, while computer vision programs are the specialized tools for the specific tasks.

Object detection, identification and tracking are some of the most fundamental usage of the computer vision. Applications, with drones, such as wind turbine inspection, powerline inspection, surveillance, search and rescue can all use object/human detection and tracking. Because of its importance in computer vision, this project is dedicated to create a program that can perform template based object detection and tracking, so as to accumulate experience in this area.

There are many methods used in object recognition or pattern recognition in general. There is texture analysis [1], which uses the objects' global properties, such as "coarseness" and "roughness" to detect object. There is Structural and Syntactic Pattern Recognition [2], the program will be given a structural description of the item to detect the object. There is also methods for Automatic Feature Selection [3], which is useful for selecting tracking features. The Kernel-based Object Tracking[4] used for tracking promises ability to cope with camera motion, partial occlusion, clutter and target scale variation, which is very useful for our goal of detecting and tracking object on a moving platform.

PROBLEM STATEMENT

The goal of this project is to create a program that can identify and track an object using a provided template. There are three main challenges for reaching the goal:

- The feature used for tracking must be rotation and scale invariant.
- The program must be able to react to object scale changes during tracking.
- The program must be able to run in real-time using a typical consumer-grade computer hardware.

The first challenge means the program cannot use the convolution template matching kind of algorithm, because, without know the orientation and scale of the object, there is no way to transform the template to search the object in the first place. Also, solving the first challenge more or less relies on the solution of the second challenge. The appropriate scaling can better isolate the object from the background, thus making the computed feature more accurate. The third challenge is to limit the computation costs.

METHODS

Detecting and tracking an object on a moving platform would require our program to be very robust to rotation and scale changes of the object in the camera frame. Thus, when choosing features to be used for tracking, strong emphasis are put on making sure those features are as rotation and scale invariant as possible. The end goal is to develop a program that can detect and track an object in the image frame using a provided object template.

Color Indexing [5]

To track an object in motion, we must use features that are reasonably unique and are rotation and scale invariant. The histogram

information is especially suited for such a task, since the histogram information should change very little if the object is rotated parallel to the plane of camera vision. If the object has consistent color pattern on all sides, then, histogram should stay invariant for all rotation motions. Object in motion often cause motion blurs in the camera image frame if the exposure is slow relative to the object motion. The motion blurs can potentially smudge some key features of the object image, but the histogram should stay unaffected. Using the method called Color Indexing [1], an object can be tracked based on its color within the image frame.

First, a template of the object to be tracked is provided to the program, the program analyzes the pixels of the template and stores the normalized histogram information onto a matrix for later reference. Then, the program uses a window that is the same size as the template to sweep through every frame and computes the normalized histogram information for each window. For each window, the program compares the histogram and compares it to that of the template and assign each window a match value. The match value, $H(I,M)$, is computed using equations shown below:

$$H(I,M) = \prod_{RGB=1}^{3} \frac{\sum_{j=1}^{n} \min(I_j, T_j)}{\sum_{j=1}^{n} T_j}$$

I's are the windowed image histogram bins and T's are the template image histogram bins. For each channel, the fraction of overlapping histogram over the template histogram is computed and, then, multiplied together to get the overall match value, $H(I,M)$. Then, a threshold value can be set to detect and track a template object within the image frame.

Modified Color Indexing

Despite being able to track a template with some rotation and scale variation, the experimental result shows that color indexing method is very sensitive to the lighting condition. Even the change due to object orientation will occasionally make program to lose track of the object. In reality, we would want to program to have consistent detection and tracking performance at various lighting conditions. To solve this problem, the RGB color space must be replaced by a color space that separates color information and brightness information. There was research focusing fast implementation of color constancy algorithm [6] for computer vision problem. Usually, the most perceptually accurate color space to use would be CIELUV, however, there is no straight forward conversion method from RGB space to Lab space without knowing certain device characteristics. Thus, it cannot be implemented for this project. Another widely used color space in image processing is Hue-Saturation-Value (HSV) color space, and we use HSV histogram to replace the RGB histogram for more robustness. The conversion from RGB space to HSV space is shown below:

$$R' = \frac{R}{255} \qquad G' = \frac{G}{255} \qquad B' = \frac{B}{255}$$

$$Cmax = \max(R', G', B')$$

$$Cmin = \min(R', G', B')$$

$$\Delta = Cmax - Cmin$$

$$H = \begin{cases} 0°, & \Delta = 0 \\ 60° \times (\frac{G' - B'}{\Delta} mod 6), & Cmax = R' \\ 60° \times (\frac{B' - R'}{\Delta} + 2), & Cmax = G' \\ 60° \times (\frac{R' - G'}{\Delta} + 4), & Cmax = B' \end{cases}$$

$$S = \begin{cases} 0, & Cmax = 0 \\ \frac{\Delta}{\mathrm{Cmax}}, & Cmax \neq 0 \end{cases}$$

$$V = Cmax$$

Instead treating each channel the same by multiplying the match value of individual channels to get the overall match value for a window coordinate, we can, now, discriminate V channel and put more emphasis on H and S channels.

$$H(I,M) = \prod_{HSV=1}^{3} \left(\frac{\sum_{j=1}^{n} \min(I_j, T_j)}{\sum_{j=1}^{n} T_j} \right)^{W}$$

, where W, is the channel match value switch value, which is between 1 and 0. To reflect the fact that we want to reduce the effect of lighting conditions having on the match value, we would make the channel switch value less than 1 for V channel.

Dynamic Scale Change

While detecting object using a provided templated, the color indexing method as described previously offers some tolerance to scale change. However, the scale difference between the object and the template could make the program less robust due to background noise. Also, if the detection window were needed for further analysis using some other methods, having a partial template match or excessive background information could make such tasks difficult. To solve this problem and improve the program performance, a dynamic scale change method is implemented. Initially, the program will scan the image trying to match windows of the image frame to the 100% scaled template. Upon finding a match to the template, the program will then iterate scales from 94% to 106% on the same detected imaged coordinate and determine that which scale produces the largest match value. By fine tuning the scale repeatedly, the detection window will shrink as the object moves further away and will expand as the object moves closer to the camera.

Trajectory Smoothing

The raw match value indicator is very noisy and sensitive to the background distraction. During the tracking, the program could be temporarily distracted by the background and shift the detection window dramatically in one frame. This tracking output is not only inaccurate, it is not physical. In a physically reasonable model, the object must move across space with some spatial continuity and smoothness. To reflect this reality, a Kalman filter [7] for three dimensional track is implemented. The Kalman filter computes 9 dynamic variable (location, velocity and acceleration). The transition matrix of the Kalman filter is define as following:

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The Kalman filter measures three variables: 2-dimentional match location and match scale.

Dynamic Windowing

With the implementation of Kalman filter, a dynamic windowing method can also be implemented to speed up the program execution. The Kalman makes prediction of the object location based on detections in previous frames. Using the predicted object location, we can limit the scanning region to be a window centered on the predicted location, instead of the entire image.

Edge Information Comparison

Since color indexing method uses only histogram information to track an object, a different object with similar color could easily trigger false positive detection and tracking. To further improve the detection and tracking accuracy, we can use the coordinate output from the modified color indexing method, and conduct feature matching between the template image and the coordinate surrounding region. Because the object is expected to move and rotate in space, edge template matching becomes unsuitable, since a slight error in rotation and scale recovery process could result in a large difference in the edge match value given the edges are thin. So, instead of using the raw edge information, three approaches are considered:

1) Edge density map
2) Edge density value
3) Edge energy value

The edge density map is used to allow more tolerance on the source-target misalignment and improve the detection and tracking accuracy. Before comparing the edge density, it is necessary to rectify the windowed frame, so that the comparing objects have the same orientation and scale. This can done by extracting certain features from the object, and based on the feature orientation, we can predict the object orientation. The approximate scale is available from implementing the dynamic scaling method. For example, strong edge direction can be used to predict orientation for a boxy object. The magnitude of the strong edge can be used to predict the scale. Once the window has been rectified, we can use a Gaussian filter to blur the edge to create the edge density map, then, conduct necessary comparison between the template and the window. However, not all objects have clearly definable feature that can be used to estimate the orientation, and the scale from the dynamic scale change is not always precise enough for edge density comparison. It is preferred to have a program that would work for more general objects with less presumptions about its shape.

A different approach is to treat the whole edge operation result holistically. By counting the number of edge elements inside the detection window and normalizing it using the size of the window, we can determine the overall edge density of the window. The edge density value is rotational invariant and, since we have implemented the dynamic scaling, the edge density value should have good tolerance to scale changes as well. Out of the three methods, edge density value is the simplest.

The edge energy method, like edge density value, is another layer of scale and rotation invariant metrics being used. Like the edge density method, we extract edge information from the window first. Using each edge pixel, the total energy of each window can be computed as following:

$$E = k \sum_{i=1}^{N} \big(e(x,y) - C(x,y)\big)^2$$

$$k = \frac{N_{temp}}{N_{window}}$$

Where, C is the centroid point of all the edge pixels in the window frame; e is the individual edge pixel; k is the normalizing factor for the number of edges in the frame, and it is computed by the ratio of total edge pixel in template and the total edge pixel in the window.

TESTING RESULTS AND DISCUSSION

TEST 1: Color Indexing Tracker Performance Test

The program is provided with following template:

Figure 1 Object Template Provide to the Program for Testing Purpose

Following is a screenshot showing the program has identified and is tracking the template:

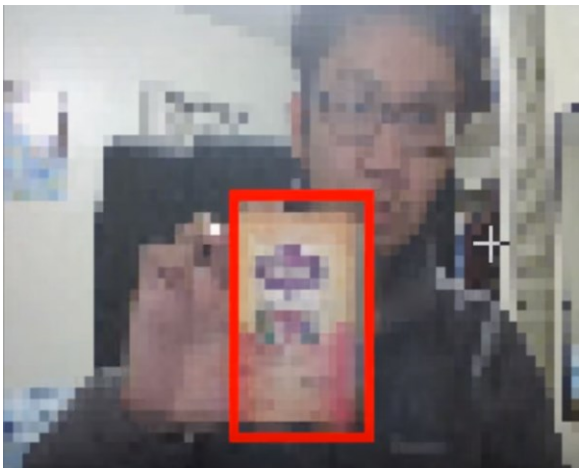

Figure 2 Color indexing program tracking an object

The image resolution has little impact on the performance of color indexing method, which is expected, since the normalized histogram should be invariant to the image resolution. This is an important characteristic we could utilize to improve the program performance later.

Another screenshot from experiment 1 shows the robustness of the color indexing method to the scale and rotation of the object:

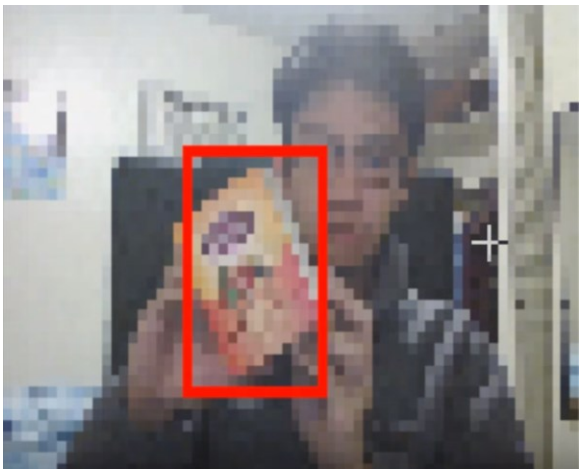

Figure 3 Color indexing program tracking an object at a smaller scale and recline position

The object is held in a reclined position at a smaller scale than the given template, and the color indexing tracker still successfully identified and tracked the object through the motion.

TEST 2: Modified Color Indexing Tracker Tested under Different Lighting Condition

Compared to the tracking performance of the color indexing method based on RGB color space, the same method based on HSV color space significantly improved the tracking consistency. Before, the angle of the camera and the lighting must be carefully adjusted for the program to work properly. Now, the program can perform reasonably well in various lighting condition. Figure 4 and Figure 5 shows the performance under different lighting conditions. In Figure 4, the desk light has been turned on, thus making the lighting environment very bright. In Figure 5, the desk light has been turned off, so, the environment is dimmer compared to the environment shown in Figure 4. Regardless, the program successfully identified and tracked the object in both lighting condition.

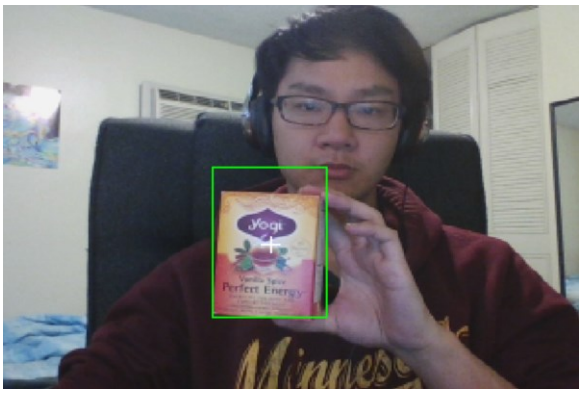

Figure 4 Object Tracking under Bright Lighting Condition

Figure 5 Program Tracking under Dim Lighting Condition

TEST 3: Dynamic Scaling Test

Dynamic scaling method was initially implemented together with the color indexing tracker base on RGB color space. Due to program being sensitive to the lighting condition, dynamic scaling method was not producing any satisfactory results, with the detection window shrinks and expands erratically and totally independent of the actual object scale. However, erratic behavior went after color indexing method was modified by changing to weighted HSV color space. The more accurate color representation provided the dynamic scaling method with sufficient match resolution, thus, enable to window to change size based on the actual object size. Figure 6 and Figure 7 shows the program tracking the object at two different scale.

Figure 6 Program Tracking the Object in an Expanded Scale

Figure 7 Program Tracking the Object in a Reduced Scale

The green window in Figure 6 and Figure 7 shows the scaled template size computed by the program. The green window in Figure 6 is much larger than the green window in Figure 7, which corresponds to the actual scale of the object is larger in Figure 6 than it is in Figure 7.

TEST 4: Trajectory Smoothing (Kalman Filter) Test

Modified color indexing and dynamic scale method provides consistent but noisy tracking. A 3-D Kalman filter was implemented to reduce the noise and make the tracking smoother. To test the smoothness of the tracking program, an object tracing code is implemented. Figure 8 shows the trajectory of tracked object moving in a circle. The trace is marked in red pixels and it is reasonable smooth.

Figure 8 Program Tracing the Object Trajectory

Experiment 5: Color Indexing False Positive Rejection using Edge Density

Color indexing methods tracks the object only based on color. Thus, it is very prone to having false positive detections. This test examines the second layer of detection verification of using edge density. If an object has been detected by the color indexing layer, a red bounding window will appear around the object. If the detection also passes the second layer of edge density test, the bounding rectangle will turn green.

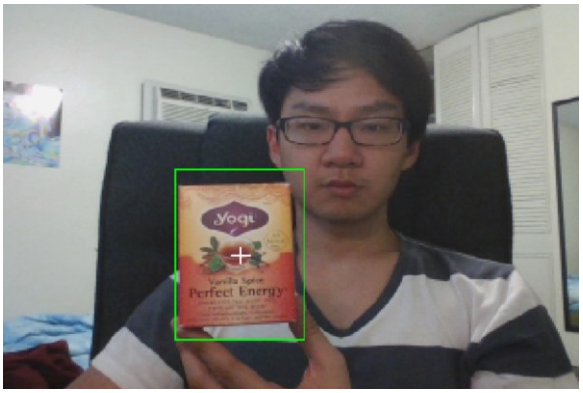

Figure 9 Program Successfully Identified the Object

In Figure 9, the program successfully identifies the object, because the windowed region not only have a good match in color to the template, but also have a similar overall edge density.

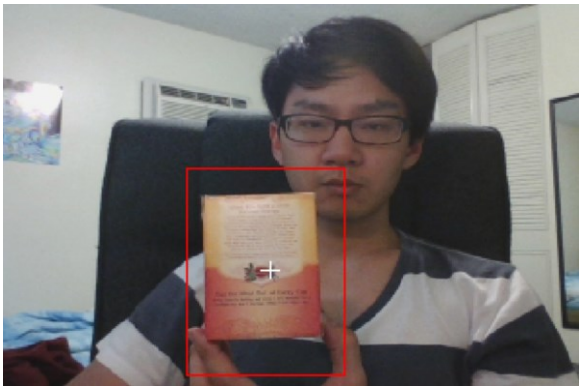

Figure 10 Program Rejects a False Positive

In Figure 10, the program successfully rejects a false positive. The side of the object has the same color as the template provided to the program, but it is not the actual template. The program sees a lower edge density in the detected region than the template, therefore, reject the detection as false positive.

When the object is moving fast across image space, the motion blurs will reduce the strength of some edges, thus causing false negative as shown in Figure 8. The factors causing this behavior is not only algorithm and speed, but also camera setting, such as exposure time and frame rate. Thus, the performance reduction caused by object moving fast enough to cause motion blur is not considered a test failure.

LESSON LEARNED

Color information are good scale and rotation invariant feature for tracking if the target appearance is known. The performance of color based tracking program could vary significantly for using different color space representation. Particularly, the typical RGB color space used in most image file blends the Chroma and Luma information together. This means that RGB color space is not suitable for applications if color constancy under various lighting condition is important. HSV color space is a much better option because it separates Chroma and Luma,

and if different channels are weighted properly, it could offer a significant performance improvement. Overall, Lab color space is the best option for color constancy, but it requires information about the camera characteristics.

The main difficulty for creating a reliable and accurate object detection program is not finding features to use for tracking, but setting appropriate detection thresholds for features or the combination of features used for tracking. A reliable and accurate object detection program is unlikely to rely on only one or two features, just like human do not identify something as a bicycle as long as two circle and a triangle are detected. Therefore, to create a good object detection and tracking program, we might have to set thresholds for many features. Such tasks quickly becomes unattainable using trial and error method because the dimensions are large and combination might be highly nonlinear. Therefore, machine learning might become useful.

When selecting features for object tracking, we sometimes rely on our intuition and judgement on the feature that we think are unique and representative. However, some cognitively strong features might actually be numerical weak features and some seemingly weak features might be very numerically strong. For example, despite being perceptually identical, the highest color match achieved between the window and the template under identical lighting condition is only less than 50%. To create a good object detection and tracking program without machine learning methods will be very difficult.

FUTURE WORK

As discussed in the previous section, machine learning is very important to make significant improved on object detection and tracking. Therefore, the future study and implementation will focus on the use of unsupervised/supervised learning to identify and track the object with greater reliability and to recognize the temporal sequential pattern.

CONCLUSION

Due to the limited time and limited man power involved in the project, the results from this project ought not to be considered as the limit of how good those involved methods could do. Instead, it should be considered as a demo or proof of concept that those methods, modified methods and concept does work, and there is room for improvement and upgrade. Also, the prospect of using machine learning to further improve the object detection and tracking program performance cannot be overstate. Compared to manual feature selection and extraction, unsupervised pattern extraction using unlabeled data can lead a more comprehensive representation of the model, thus, making detection and tracking far more accurate.

REFERENCE

[1] Tuceryan, Mihran and Jain, Anil K, “Texture Analysis.” The Handbook of Pattern Recognition and Computer Vision (2nd Edition) (1998): 207-248

[2] Bunke, Horst. Structural and syntactic pattern recognition, Handbook of Pattern Recognition and Computer Vision. World Scientific. (1993) pp. 163–209.

[3] Siedlecki, Wojciech and Sklansky, Jack. “On Automatic Feature Selection”, International Journal of Pattern Recognition Artificial Intelligence. 02 (1988) 197

[4] D. Comaniciu, V. Ramesh and P. Meer. “Kernel based object tracking”, IEEE Trans. Pattern Anal. Machine Intell., 25, 564-577, 2003.

[5] Swain, Michael J., and Dana H. Ballard. "Color indexing." International journal of computer vision 7.1 (1991): 11-32.

[6] Jean-Michel Morel ; Ana B. Petro ; Catalina Sbert; Fast implementation of color constancy algorithms. Proc. SPIE 7241, Color Imaging XIV: Displaying, Processing, Hardcopy, and Applications, 724106 (January 19, 2009)

[7] Kalman, R. E. (1960). "A New Approach to Linear Filtering and Prediction Problems". *Journal of Basic Engineering* 82: 35