CO352 - Computer Graphics Mini Project

Solving Waldo's Puzzle - Progress 1

Overview

The project's aim is to solve the Waldo's puzzle, i.e. to find the position of Waldo in an image filled with lots of different characters.

Things Implemented

1. Ran the code for linear matching methods, and successfully found Waldo in images.
2. Collected datasets for Waldo's puzzles and Waldo templates.

Matching Method

To identify the matching area, we have to *compare* the template image against the source image by sliding it.By sliding, we mean moving the patch one pixel at a time (left to right, up to down). At each location, a metric is calculated so it represents how "good" or "bad" the match at that location is (or how similar the patch is to that particular area of the source image).For each location of **T** over **I**, you *store* the metric in the *result matrix* **(R)**. Each location (x,y) in **R** contains the match metric.The brightest locations indicate the highest matches.

OpenCV implements Template matching in the function matchTemplate,

void matchTemplate(Input image, Input templ, Output result, int method)

Parameters:

- **image** – Image where the search is running. It must be 8-bit or 32-bit floating-point.
- **templ** – Searched template. It must be not greater than the source image and have the same data type.
- **result** – Map of comparison results. It must be single-channel 32-bit floating-point. If image is $W \times H$ and templ is $w \times h$, then result is $(W-w+1) \times (H-h+1)$.
- **method** – Parameter specifying the comparison method

Methods available in openCV

1. CV_TM_SQDIFF

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

2. CV_TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

3. CV_TM_CCORR

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

4. CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

5. CV_TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))$$

Where $T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$
$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$

6. CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

we use the function minMaxLoc to locate the highest value (or lower, depending of the type of matching method) in the *R* matrix.

CV_SQDIFF and CV_SQDIFF_NORMED the best match are the lowest values.ie Match location is minLoc

For all the others, higher values represent better matches.ie Match location is maxLoc

Things to be done

1. Train model to learn Waldo's physical features.
2. Perform search for Waldo and find his position using the machine learned(trained) model.

Technologies Used

1. openCV - C++ library
2. tensorflow

References

1. The system of face detection based on openCV
   -http://ieeexplore.ieee.org/abstract/document/6242980/
2. Face Detection and Tracking using OpenCV
   -http://www.thesij.com/papers/CNCE/2013/July-August/CNCE-0103540102.pdf
3. Real Time Face Detection using OpenCV
   -http://www.iraj.in/journal/journal_file/journal_pdf/4-54-140014639841-44.pdf