Template Matching

Lecture #11

February 12, 2018

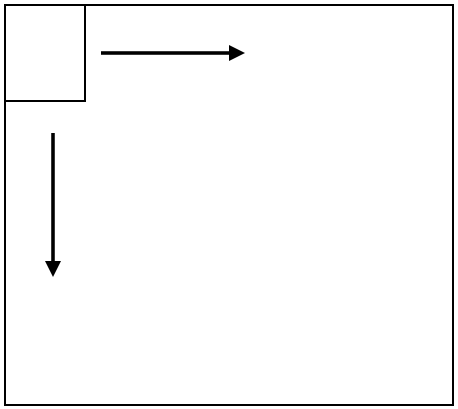
Colorado State University

Find similar patterns in a larger image

- The image above is a small piece of the image to the right. But from where?

Brute-Force Translation Invariance

To find a small image in a large one, "slide" the small one across the large, computing Pearson's correlation at every possible position.



Statistical Cross-Correlation

- The process of "slide & correlate" is called cross-correlation
- Complexity is O(nm)
  - N = # of pixels in image (w×h)
  - M = # of pixels in the template (w×h)
- Highly parallel (every position can be computed independently)
- Still sensitive to
  - Rotation
    - in-plane
    - out-of-plane
  - Scale
  - Perspective

Computing Cross-Correlation

- In cross-correlation, the mask is correlated repeatedly to image windows
  - zero-mean & unit length the mask
  - zero-mean & unit length the image
  - compute the sliding dot product

This is *almost* convolving the image with the mask

Naming conventions

- In Engineering, convolving a normalized mask with the source image is called correlation
  - Is this exactly the same as Pearson's correlation?
  - Why or why not?
- This is the most common definition of correlation in image processing texts

Application: Tracking

- Cut out a picture of a target from the first frame of a video
  - Use it as a template /mask
- Correlate the target in the following frames
  - Find the location with the highest correlation
- Improvement:
  - update target with each new frame

Application: Tracking

Application: Mosaicing

- Take several, overlapping images from a translating camera
  - Camera cannot move along optical axis
- Correlate the whole images to each other
  - Find location where they match the best
  - Stitch them into a single, larger image

Mosaicing (II)

Image 1

Image 2

Image 3

In OpenCV

docs.opencv.org/2.4/modules/imgproc/doc/obj

Object Detection — OpenCV 2.4.13.5 documentation

OpenCV

Search

Hide Search Matches

Table Of Contents

Object Detection
- matchTemplate

Previous topic

Feature Detection

Next topic

highgui. High-level GUI and Media I/O

This Page

Show Source

matchTemplate

Compares a template against overlapped image regions.

C++: void **matchTemplate**(InputArray **image**, InputArray **templ**, OutputArray **result**, int **method**)

Python: cv2.**matchTemplate**(image, templ, method[, result]) → result

C: void **cvMatchTemplate**(const CvArr* **image**, const CvArr* **templ**, CvArr* **result**, int **method**)

Python: cv.**MatchTemplate**(image, templ, result, method) → None

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 (see below).

The function slides through `image`, compares the overlapped patches of size $w \times h$ against

Colorado State University

Use OpenCV Tutorial

docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/te

Template Matching — OpenCV 2.4.13.5 documentation

- **Downloadable code**: Click here
- **Code at glance:**

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdio.h>

using namespace std;
using namespace cv;

/// Global Variables
Mat img; Mat templ; Mat result;
char* image_window = "Source Image";
char* result_window = "Result window";

int match_method;
int max_Trackbar = 5;

/// Function Headers
void MatchingMethod( int, void* );

/** @function main */
int main( int argc, char** argv )
{
  /// Load image and template
  img = imread( argv[1], 1 );
  templ = imread( argv[2], 1 );

  /// Create windows
  namedWindow( image_window, CV_WINDOW_AUTOSIZE );
```

Our Example 8