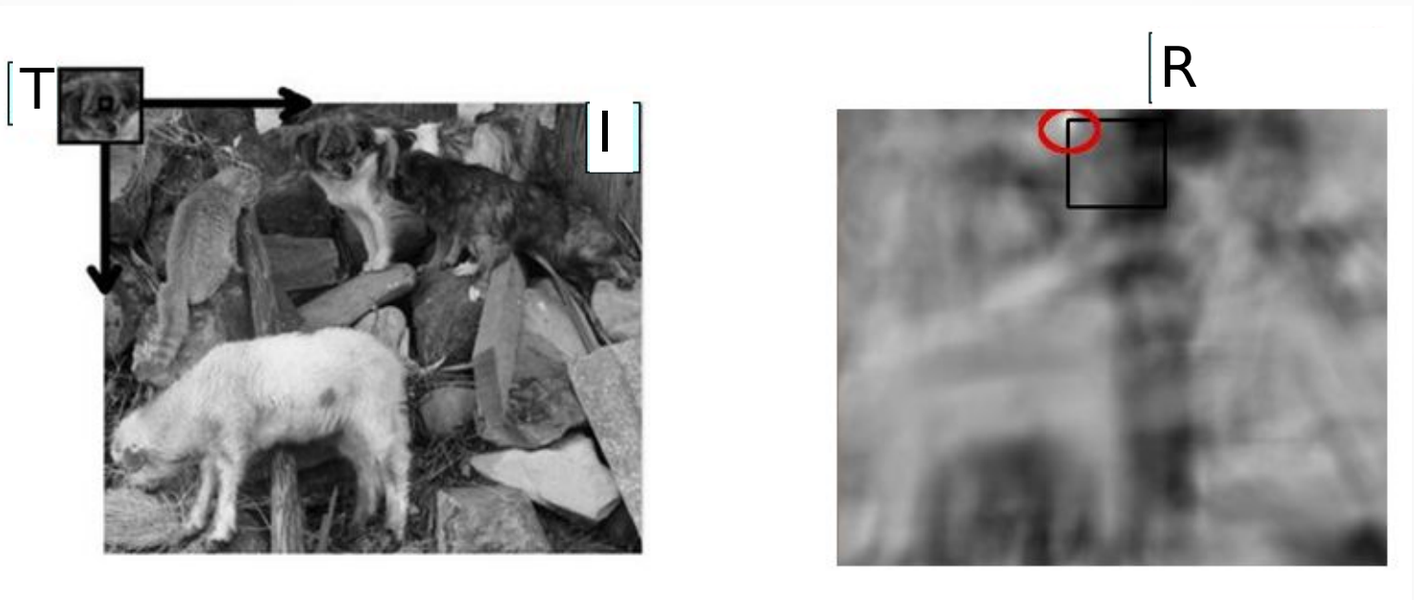
Template Matching

By Noah Murad
CS 889

Problem

- Given an image *I[x,y]* find an object represented by a template *T[x,y]* by sliding *T* over *I* and using a *matching function* to get *R[x,y]*
- Nearly as simple as convolution



Matching Functions

- Difference Functions (lower *R[x,y]* a better match)
  - Square Difference
- Similarity Functions (higher *R[x,y]* a better match)
  - Correlation
  - Correlation Coefficient
- Normalized Variants
  - The three functions above have normalized variants which alter their behaviour and make *R[x,y]* fit in the range [0,1]
- Each of the 6 functions have their own uses
  - I am yet to discover them all
- Details on the math here

What do I do with R[x,y]?

- Find the pixel containing the min/max value
  - This is the best match of the template in the image
    - According to the matching function
- Threshold *R* to find multiple matches
  - Only look at local maxima/minima to eliminate multiple matches on the same instance
    - Check the code sample under “Edge-Based Template Matching” at line 25 a more intuitive explanation

Strengths and Limitations

- Strengths
  - Simple to implement/understand
    - Clear white box – easy to debug
  - Fast (O(len(I)*len(T))
  - Great for synthetic scenes
- Weaknesses
  - Not rotation or scale invariant*
    - See next slides
  - Usually too brittle on natural scenes

Enhancements – Rotation and Scale Invariance

- We can hack template matching to be rotation and scale invariant
- 1. resize and/or rotate the source image
  - The more sizes/rotations, the more accuracy
- 2. template match on each variation of the source
  - Store the best match, and its corresponding rotation/scale
- 3. inversely apply the best match’s rotation/scale to the final position and size
- *see the code sample for an implementation

Enhancements – Preprocessing for Template Matching

- Use an edge detector before performing template matching
  - Speeds up template matching time
  - Brightness/color invariant
  - Can be much more robust in many situations

Applications

- Preprocessing
- Finding features in synthetic scenes
  - Such as user interfaces
- Whatever you can image

That's all

- Thanks for listening
- The code sample has a fun game at the end