Template Matching : Multiple Approaches

- Main idea of this "section" is to showcase a couple of ideas.
  - sometimes image is interpreted to be a function (continuous, or more so, continuously differentiable). Techniques for continuous functions are then used and translated back to the discrete image.
  - within the continuous framework, the image will be placed into an optimization problem whose solution manipulates the image or extracts information from the image.

Version 1: direct, global search

consider an image I and a template T (of smaller size). the goal is to find where T is within the larger image I.

$I : D \to \mathbb{R}$ ← grayscale image over domain $D = [0, W-1] \times [0, H-1]$ (width, height)

$T : D_T \to \mathbb{R}$ ← grayscale image patch where $D_T = [-\frac{w}{2}, \frac{w}{2}] \times [-\frac{h}{2}, \frac{h}{2}]$ (width, height)

one way to find the location is to ~~guess a location and~~ define a ~~or the~~ template matching score function

$$\mathcal{E}(\vec{u}) = \iint_{D_T} [\, I(\vec{x}+\vec{u}) - T(\vec{x}) \,]^2 \, d\vec{x}$$

The score function or energy is known as the sum of squared differences (for being precisely that).

Searching over all possible locations $\vec{u} \in D$ and finding the lowest score gives the best possible estimate of the template location. In math speak:

$$\arg\min_{\vec{u}\in D} \mathcal{E}(\vec{u}) = \arg\min_{\vec{u}\in D} \iint_{D_T} [I(\vec{x}+\vec{u}) - T(\vec{x})]^2 d\vec{x}$$

↑ want the argument $\vec{u}$ that minimizes the energy $\mathcal{E}$.

the first version here tries to ~~evalut~~ evaluate the energy everywhere. First, ~~convert the continuous integrals to summations~~

~~$\mathcal{E}(\vec{u}) = \sum\sum [I(\vec{x}+\vec{u})$~~

$$\mathcal{E}(\vec{u}) = \iint_{D_T} I^2(\vec{x}+\vec{u})\, d\vec{x} - 2\iint_{D_T} I(\vec{x}+\vec{u}) \cdot T(\vec{x})\, d\vec{x} + \iint_{D_T} T(\vec{x})\, d\vec{x}$$

- $\iint_{D_T} I^2(\vec{x}+\vec{u})\, d\vec{x}$: can be evaluated for all possible $\vec{u}$
- $\iint_{D_T} I(\vec{x}+\vec{u}) \cdot T(\vec{x})\, d\vec{x}$: actually equal to the cross correlation of the template w/ the image when evaluated for all $\vec{u}$.
- $\iint_{D_T} T(\vec{x})\, d\vec{x}$: constant since template does not change.

Implementation in Matlab

first term can be evaluated for all $\vec{u}$: `I.^2`

then integral ~~[illegible]~~ over window can be evaluated using an imfilter kernel of all ones.

```
Isq = imfilter( I.^2, ones(h,w))
```

↳ height & width of template.

second term is cross-correlation

```
Ixc = xcorr2(I, T)
```

third term is constant

```
Tsq = sum(sum( T.^2));
```

or...

```
    = dot(T(:), T(:));
```

or...

lots of ways to compute.

then

```
E = Isq - 2.*Ixc + Tsq;
```

gives energy for all possible $\vec{u} \in D$.
next step is to find the minimizing $\vec{u}$.
just look for the index into the smallest value of E.

Version 2: Skipping steps ... Cross-correlation

Examine the energy once more:

$$\xi(\vec{u}) = \iint_{D_T} I^2(\vec{x}+\vec{u})\,d\vec{x} - 2\iint_{D_T} I(\vec{x}+\vec{u})T(\vec{x})\,d\vec{x} + \iint_{D_T} T^2(\vec{x})\,d\vec{x}$$

VARIES W/ $\vec{u}$ BUT DOES NOT DEPEND ON $T(\vec{x})$. ALWAYS POSITIVE (OR ZERO)

VARIES WITH $T(\vec{x})$. NOT CONSTANT. CAN BE POSITIVE OR NEGATIVE.

CONSTANT AND POSITIVE! (OR ZERO)

note that the first and third terms only add to the energy. only the second term can lower the energy.

Hey! Maybe looking only at the second term will work! What if we ~~only~~ tried to maximize this term only?

New problem is:

$$\arg\max_{\vec{u}\in D} \iint_{D_T} I(\vec{x}+\vec{u})T(\vec{x})\,d\vec{x}$$

MATLAB IMPLEMENTATION:

```
Ixc = xcorr2(I,T);
```

find indices into max value of Ixc.

Extension: the cross-correlation can have false maxima if the image has "bright" spots. So, often the normalized cross correlation is used:

$$\frac{\iint_{D_T} I(\vec{x}+\vec{u})\, T(\vec{x})\, d\vec{x}}{\left(\iint_{D_T} I^2(\vec{x}+\vec{u})\, d\vec{x}\right)^{1/2} \left(\iint_{D_T} T^2(\vec{x})\, d\vec{x}\right)^{1/2}}$$

more robust to illumination differences between I & T.

MATLAB IMPLEMENTATION:

```
nIxc = normxcorr2( T, I )
```

image second

template first

values always lie between $[-1, 1]$.

Version 3: Gradient Descent.

If initial estimate of solution is known, then can search nearby.

Better yet, can search in direction that minimizes the energy $\mathcal{E}$.

↳ such a search tactic is embodied by the following gradient descent approach.

if $\mathcal{E} : D \to \mathbb{R}$ gives energy/score of the guess $\vec{u}$, then

$D\mathcal{E}(\vec{u}) \cdot \delta\vec{u}$ gives change in score in the direction $\delta\vec{u}$.

↑ the differential

sometimes written as $\nabla\mathcal{E}(u) \cdot \delta\vec{u}$

↑ dot product

↑ gradient

↑ direction of change.

to minimize $\mathcal{E}$, $\delta\vec{u}$ should cause a negative change,

$$D\mathcal{E}(\vec{u}) \cdot \delta\vec{u} < 0$$

the best vector to use is the gradient vector (negative)

if $\delta\vec{u} = -\nabla\mathcal{E}(\vec{u})$, ~~then~~ ~~$D\mathcal{E}(\vec{u}) \cdot \delta\vec{u} = \nabla\mathcal{E}(u) \cdot (-\nabla\mathcal{E}(u))$~~

~~then~~ then $D\mathcal{E}(\vec{u}) \cdot \delta\vec{u} = \nabla\mathcal{E}(u) \cdot (-\nabla\mathcal{E}(u))$

$$= -\nabla\mathcal{E}(u) \cdot \nabla\mathcal{E}(u) < 0$$

OK, but what is $D\mathcal{E}(\vec{u})$??? or, equivalently, $\nabla\mathcal{E}(\vec{u})$???

Let's work it out

$$D\mathcal{E}(\vec{u}) = \frac{\partial \mathcal{E}(\vec{u})}{\partial \vec{u}} = \frac{\partial}{\partial \vec{u}} \iint_{D_T} \left[I(\vec{x}+\vec{u}) - T(\vec{x}) \right]^2 d\vec{x}$$

$$= \iint_{D_T} \frac{\partial}{\partial \vec{u}} \left[I(\vec{x}+\vec{u}) - T(\vec{x}) \right]^2 d\vec{x}$$

$$= \iint_{D_T} 2 \left[I(\vec{x}+\vec{u}) - T(\vec{x}) \right] \cdot \underbrace{DI(\vec{x}+\vec{u})}_{\| \; \frac{\partial I(\vec{x}+\vec{u})}{\partial \vec{u}}} d\vec{x}$$

$\Rightarrow$

$$\nabla\mathcal{E}(\vec{u}) = 2 \iint_{D_T} \left[I(\vec{x}+\vec{u}) - T(\vec{x}) \right] \cdot \nabla I(\vec{x}+\vec{u}) \, d\vec{x}$$

so,

$$\delta\vec{u} = -2 \iint_{D_T} \left[I(\vec{x}+\vec{u}) - T(\vec{x}) \right] \cdot \nabla I(\vec{x}+\vec{u}) \, d\vec{x}$$

thus update to initial guess is:

$$\vec{u} = \vec{u} + dt \cdot \delta\vec{u} \; ;$$

↑ necessary because $\delta\vec{u}$ is a direction that decreases $\mathcal{E}$, but may be of incorrect magnitude.

the dt rescales vector $\delta\vec{u}$ to give a proper update. It must be tuned to the problem.

MATLAB IMPLEMENTATION (ROUGH SKETCH):

1. need to extract image window centered at $\vec{u}$ of dimensions ~~(w x h)~~ $(w+2) \times (h+2)$. Use imcrop ← for pixel level accuracy
   interp2 ← for sub pixel level accuracy

   ↑ buffer needed to compute gradients properly.

2. compute gradients of the image gx, gy. (use convolution kernel)
3. throw away buffer region to get (w x h) matrices.
4. compute difference image dI = Iwindow − T ;

   ↑ windowed image ↑ template.

5. $\delta u = [\text{sum}(\text{sum}(dI .* gx)); \quad \text{sum}(\text{sum}(dI .* gy))];$

   ↑ double sum is like double integral.

6. update $\vec{u}$, $u = u + dt \cdot \delta u$;
7. repeat until $\vec{u}$ does not change or until max number of allowed repeats hit.

Version 4: Gradient Descent Using Quadratic Approximation.

Rather than gradient descend on ξ, a linearized version will be used.

consider $\xi(\vec{u}_0 + \delta\vec{u}) = \iint_{D_T} [I(\vec{x}+\vec{u}_0+\delta\vec{u}) - T(\vec{x})]^2 d\vec{x}$

a linearization of the integrand uses:

$$I(\vec{x}+\vec{u}_0+\delta\vec{u}) = I(\vec{x}+\vec{u}_0) + DI(\vec{x}+\vec{u}_0)\cdot\delta\vec{u}$$

$\Rightarrow$

$$\xi(\vec{u}_0+\delta\vec{u}) \approx \iint_{D_T} [I(\vec{x}+\vec{u}_0) + DI(\vec{x}+\vec{u}_0)\cdot\delta\vec{u} - T(\vec{x})]^2 d\vec{x}$$

$$\approx \iint_{D_T} [(I(\vec{x}+\vec{u}_0)-T(\vec{x}))^2 + 2(I(\vec{x}+\vec{u}_0)-T(\vec{x}))DI(\vec{x}+\vec{u}_0)\delta\vec{u} + (DI(\vec{x}+\vec{u}_0)\,\delta\vec{u})^2]\, d\vec{x}$$

$$\approx \iint_{D_T} (I(\vec{x}+\vec{u}_0)-T(\vec{x}))^2 d\vec{x} - 2\left(\iint_{D_T} (I(\vec{x}+\vec{u}_0)-T(\vec{x}))\, DI(\vec{x}+\vec{u}_0)\, d\vec{x}\right)\cdot\delta\vec{u} + \iint_{D_T} (DI(\vec{x}+\vec{u}_0)\,\delta\vec{u})^2 d\vec{x}$$

↳ Ah, but $(DI(\vec{x}+\vec{u}_0)\cdot\delta\vec{u})^2$ (both scalar)

$= (DI(\vec{x}+\vec{u}_0)\cdot\delta\vec{u})(DI(\vec{x}+\vec{u}_0)\cdot\delta\vec{u})$

$\square$

$$= \left(DI(\vec{x}+\vec{u}_0)\cdot\delta\vec{u}\right)^T \left(DI(\vec{x}+\vec{u}_0)\cdot\delta\vec{u}\right)$$

$$= \delta\vec{u}^T\, DI^T(\vec{x}+\vec{u}_0)\, DI(\vec{x}+\vec{u}_0)\, \delta\vec{u}$$

$$= \delta\vec{u}^T \left(DI^T(\vec{x}+\vec{u}_0)\, DI(\vec{x}+\vec{u}_0)\right)\delta\vec{u}$$

$$= \delta\vec{u}^T \left(\nabla I(\vec{x}+\vec{u}_0)\cdot \nabla I^T(\vec{x}+\vec{u}_0)\right)\delta\vec{u}$$

↑ A matrix 2×2

where I used the fact that $DI(\vec{x}) = \nabla I^T(\vec{x})$

differential & gradient are transposes of each other.

⇒ written using gradients

$$E(\vec{u}+\delta\vec{u}) \approx \iint_{D_T} \left(I(\vec{x}+\vec{u}_0) - T(\vec{x})\right)^2 d\vec{x}$$

$$- 2\left(\iint_{D_T} \left(I(\vec{x}+\vec{u}_0) - T(\vec{x})\right) \nabla I(\vec{x}+\vec{u}_0) d\vec{x}\right)\delta\vec{u}$$

$$+ \delta\vec{u}^T \left(\iint_{D_T} \nabla I(\vec{x}+\vec{u}_0)\, \nabla I^T(\vec{x}+\vec{u}_0)\, dx \right)\delta\vec{u}$$

note that $E(\vec{u}_0+\delta\vec{u})$ looks like

$$C - 2B\cdot\delta\vec{u} + \delta\vec{u}^T A\,\delta\vec{u} -$$

$$= C - 2B\cdot\delta\vec{u} + (A\delta\vec{u})\cdot\delta\vec{u}$$

which is a quadratic energy function.
The minimizer can be found by setting the "derivative" to zero

⇒

$$-2B + 2A\,\delta\vec{u} = 0$$

⇒

$$\delta\vec{u} = A^{-1}B$$

so,

$$\delta\vec{u} = \left[\iint_{D_T} \nabla I(\vec{x}+\vec{u}_0)\, \nabla I^T(\vec{x}+\vec{u}_0)\, dx \right]^{-1} \left[\iint_{D_T} \left[I(\vec{x}+\vec{u}_0) - T(\vec{x}) \right] \nabla I(\vec{x}+\vec{u}_0)\, dx \right]$$

update is

$$\vec{u} = \vec{u}_0 + \delta\vec{u} .$$

repeat with $\vec{u}_0$ set to $\vec{u}$. (the updated version)

MATLAB IMPLEMENTATION.

initial steps similar to version 3, however

~~if~~ gradI = [gx(:) , gy(:)] ;

then

~~A = transpose(gradI)~~

A = transpose (gradI) • gradI ;

B = transpose (transpose (gradI) • dI(:)) ;

automatically does the double sum (e.g. double integral)

and

du = A^-1 B ;

↳ if A not invertible, then image region is crappy. (insufficient variation)