Intro to Template Matching
and the Lucas-Kanade Method
Appearance-Based Tracking

current frame + previous location

appearance model
(e.g. image template, or color; intensity; edge histograms)

Mode-Seeking
(e.g. mean-shift; Lucas-Kanade; particle filtering)

likelihood over object location

current location
Tie In with Bayesian Tracking

If posterior on object location has a single dominant mode close to predicted object location, we can use hill-climbing methods to find the mode, even without computing the posterior distribution explicitly.
Basic Template Matching

- Assumptions:
  - a snapshot of object from first frame can be used to describe appearance
  - Object will look nearly identical new image
  - Movement is nearly pure 2D translation

The last two are very restrictive. We will relax them later on.
Template Matching

• Is a “search” problem:
  – Given an intensity patch element in the left image, search for the corresponding patch in the right image.
  – We will typically need geometric constraints to reduce the size of the search space
  – But for now, we focus on the matching function
Correlation-based Algorithms

Elements to be matched are image patches of fixed size

Task: what is the corresponding patch in a second image?
Correlation-based Algorithms

Task: what is the corresponding patch in a second image?

1) Need an appearance similarity function.

2) Need a search strategy to find location with highest similarity. Simplest (but least efficient) approach is exhaustive search.
Comparing Windows:

Some possible measures:

\[
\begin{align*}
\text{SSD} & = \sum_{[i,j] \in R} (f(i, j) - g(i, j))^2 \\
C_{fg} & = \sum_{[i,j] \in R} f(i, j)g(i, j)
\end{align*}
\]
Correlation $C_{fg}$

$$C_{fg} = \sum_{[i,j] \in R} f(i,j)g(i,j)$$

If we are doing exhaustive search over all image patches in the second image, this becomes cross-correlation of a template with an image.
Example

Stereo pair of the “El Capitan” formation from the NASA Mars rover mission.

Image 1

Image 2

These slides use a stereo pair, but the basic concepts are the same for matching templates across time.
Example

Image 1

Template (image patch)
Example: Cross-correlation

score = imfilter(image2,tmpl)

Highest score

Correct match

Score around correct match
Example: Cross-correlation

Note that score image looks a lot like a blurry version of image 2.

This clues us in to the problem with straight correlation with an image template.
Problem with Correlation of Raw Image Templates

Consider correlation of template with an image of constant grey value:

\[
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
  g & h & i \\
\end{array}
\]

\[
\begin{array}{ccc}
  v & v & v \\
  v & v & v \\
  v & v & v \\
\end{array}
\]

Result: \( v \cdot (a + b + c + d + e + f + g + h + i) \)
Problem with Correlation of Raw Image Templates

Now consider correlation with a constant image that is twice as bright.

\[
\begin{array}{c|c|c}
  a & b & c \\
  \hline
  d & e & f \\
  \hline
  g & h & i \\
\end{array}
\times
\begin{array}{c|c|c}
  2v & 2v & 2v \\
  \hline
  2v & 2v & 2v \\
  \hline
  2v & 2v & 2v \\
\end{array}
\]

Result: \(2v \times (a+b+c+d+e+f+g+h+i)\)

\[> v \times (a+b+c+d+e+f+g+h+i)\]

Larger score, regardless of what the template is!
Solution

Subtract off the mean value of the template.

In this way, the correlation score is higher only when darker parts of the template overlap darker parts of the image, and brighter parts of the template overlap brighter parts of the image.
Correlation, zero-mean template

Better! But highest score is still not the correct match.
Note: highest score IS best within local neighborhood of correct match.
“SSD” or “block matching” (Sum of Squared Differences)

\[
\sum_{[i,j] \in R} (f(i, j) - g(i, j))^2
\]

1) The most popular matching score.
2) We will use it for deriving the Lucas-Kanade method.
3) Trucco & Verri (486 textbook) claims it works better than cross-correlation.
Relation between SSD and Correlation

\[ SSD = \sum_{[i,j] \in R} (f - g)^2 \]

\[ = \sum_{[i,j] \in R} f^2 + \sum_{[i,j] \in R} g^2 - 2 \sum_{[i,j] \in R} fg \]

\[ C_{fg} = \sum_{[i,j] \in R} f(i,j)g(i,j) \]

Correlation!
Best match (highest score) in image coincides with correct match in this case!
Handling Intensity Changes

Intensity Changes:

• the camera taking the second image might have different intensity response characteristics than the camera taking the first image

• Illumination in the scene could change

• The camera might have auto-gain control set, so that it’s response changes as it moves through the scene.
Handling Intensity Changes:

- One approach is to estimate the change in intensity and compensate for it (e.g. “Background Estimation under Rapid Gain Change in Thermal Imagery”, Yalcin, Collins and Hebert, 2nd IEEE Workshop on Object Tracking and Classification in and Beyond the Visible Spectrum (OTCBVS'05), June, 2005)

- The second approach is to use a normalized matching function that is invariant to intensity changes. This is the one we will be discussing now.
Intensity Normalization

• When a scene is imaged by different sensors, or under different illumination intensities, both the SSD and the $C_{fg}$ can be large for windows representing the same area in the scene!

• A solution is to normalize the pixels in the windows before comparing them by subtracting the mean of the patch intensities and dividing by the std.dev.

\[
\hat{f} = \frac{f - \bar{f}}{\sqrt{\sum (f - \bar{f})^2}} \quad \hat{g} = \frac{g - \bar{g}}{\sqrt{\sum (g - \bar{g})^2}}
\]
Normalized Cross Correlation

Highest score also coincides with correct match. Also, looks like less chances of getting a wrong match.
Normalized Cross Correlation

Important point about NCC:
Score values range from 1 (perfect match) to -1 (completely anti-correlated)

Intuition: treating the normalized patches as vectors, we see they are unit vectors. Therefore, correlation becomes dot product of unit vectors, and thus must range between -1 and 1.
Voluntary Exercise

Let $f$ be an image patch, and let $g$ be the same patch except with grey values modified by a multiplicative intensity gain factor and an additive intensity offset:

$$g = \text{gain} \times f + \text{offset}$$

Consider the normalized patches:

$$\hat{f} = \frac{f - \bar{f}}{\sqrt{\sum (f - \bar{f})^2}} \quad \hat{g} = \frac{g - \bar{g}}{\sqrt{\sum (g - \bar{g})^2}}$$

Show that: $\hat{f} = \hat{g}$
Some Practical Issues

- Object shape might not be well described by a scanline-oriented bounding rectangle

We end up including lots of background pixels in our foreground template
Some Practical Issues

• One solution: use a Gaussian windowing function to weight pixels on the object more highly (and weight background pixels zero)

Works best for compact objects like vehicles.
Some Practical Issues

Problem: Don’t want to search whole image. 
Solution: bound search radius for object based on how far it can move between frames.

Need some estimate of object motion (and camera motion if camera is moving). Works best over short time periods.
Correlation-based Tracker

Properties:

• Correlation of normalized template
• Use Gaussian windowing function (computed from user supplied box in first frame)
• Search window for object centered at previous object location (best for small motion)
Normalized Correlation, Fixed Template

Failure mode: distraction by background clutter
Normalized Correlation, Fixed Template

Failure mode: Unmodeled Appearance Change
Naive Approach to Handle Change

• One approach to handle changing appearance over time is adaptive template update
• One you find location of object in a new frame, just extract a new template, centered at that location

• What is the potential problem?
Normalized Correlation, Adaptive Template

Here our results are no better than before.
Normalized Correlation, Adaptive Template

Here the result is even worse than before!
Tracking via Gradient Descent
Motivation

• Want a more efficient method than explicit search over some large window
• If we have a good estimate of object position already, we can efficiently refine it using gradient descent.

• Assumption: Our estimate of position must be very close to where the object actually is! (however, we can relax this using multi-scale techniques - image pyramids)
Math Example : 1D Gradient

Consider function $f(x) = 100 - 0.5 \times x^2$
Math Example: 1D Gradient

Consider function \( f(x) = 100 - 0.5 \times x^2 \)

Gradient is \( \frac{df(x)}{dx} = -2 \times 0.5 \times x = -x \)

Geometric interpretation:
gradient at \( x_0 \) is slope of tangent line to curve at point \( x_0 \)

\[
\begin{align*}
\Delta y & \quad \text{slope} = \frac{\Delta y}{\Delta x} \\
& = \left. \frac{df(x)}{dx} \right|_{x_0}
\end{align*}
\]
Math Example: 1D Gradient

\[ f(x) = 100 - 0.5 \times x^2 \quad df(x)/dx = -x \]

\[ x_0 = -2 \]

\[ \text{grad} = \text{slope} = 2 \]
Math Example: 1D Gradient

\[ f(x) = 100 - 0.5 \times x^2 \quad \text{df(x)/dx} = -x \]
Math Example: 1D Gradient

\[ f(x) = 100 - 0.5 \times x^2 \quad \text{df}(x)/\text{dx} = -x \]

Gradients on this side of peak are positive

Gradients on this side of peak are negative

Note: Sign of gradient at point tells you what direction to go to travel “uphill”
**Math Example: 2D Gradient**

\[ f(x,y) = 100 - 0.5 * x^2 - 0.5 * y^2 \]

\[ \frac{df(x,y)}{dx} = -x \quad \frac{df(x,y)}{dy} = -y \]

Gradient = \[ [\frac{df(x,y)}{dx} , \frac{df(x,y)}{dy}] = [-x , -y] \]

Gradient is vector of partial derivs wrt x and y axes
Math Example: 2D Gradient

\[ f(x,y) = 100 - 0.5 \times x^2 - 0.5 \times y^2 \]

Gradient = \[ \frac{df(x,y)}{dx}, \frac{df(x,y)}{dy} \] = \[ -x, -y \]

Plotted as a vector field, the gradient vector at each pixel points “uphill”.

The gradient indicates the direction of steepest ascent.

The gradient is 0 at the peak (also at any flat spots, and local minima,…but there are none of those for this function)
Math Example: 2D Gradient

\[ f(x,y) = 100 - 0.5 \times x^2 - 0.5 \times y^2 \]

Gradient = \[ \frac{df(x,y)}{dx} , \frac{df(x,y)}{dy} \] = \[ -x , -y \]

Let \( g = [g_x, g_y] \) be the gradient vector at point/pixel \( (x_0,y_0) \)

Vector \( g \) points uphill
(direction of steepest ascent)

Vector \( -g \) points downhill
(direction of steepest descent)

Vector \[ [g_y, -g_x] \] is perpendicular, and denotes direction of constant elevation. i.e. normal to contour line passing through point \( (x_0,y_0) \)
Math Example: 2D Gradient

\[ f(x,y) = 100 - 0.5 \times x^2 - 0.5 \times y^2 \]

Gradient = \[ \frac{df(x,y)}{dx} , \frac{df(x,y)}{dy} \] = \[ -x , -y \]

And so on for all points
What is Relevance of This?

consider the SSD error function

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - T(x,y)]^2$$

This tells how match score changes if you shift the template by \([u,v]\)
What is Relevance of This?

In locality of correct match, the function $E(u,v)$ looks a little bit like:

Actually, it looks like an upside down version of this since we are looking for a minimum.

Therefore, we should be able to use gradient descent to find the best offset $(u,v)$ of the template in the new frame **without explicitly computing the whole correlation surface!**
Review: Numerical Derivatives
See also Trucco&Verri, Appendix A.2

Taylor Series expansion

\[ f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + O(h^4) \]

Manipulate:

\[ f(x+h) - f(x) = hf'(x) + \frac{1}{2}h^2 f''(x) + O(h^3) \]

\[ \frac{f(x+h) - f(x)}{h} = f'(x) + O(h) \]

Finite forward difference
Numerical Derivatives

See also T&V, Appendix A.2

Taylor Series expansion

\[ f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{3!}h^3 f'''(x) + O(h^4) \]

Manipulate:

\[ f(x) - f(x-h) = hf'(x) - \frac{1}{2}h^2 f''(x) + O(h^3) \]

\[ \frac{f(x) - f(x-h)}{h} = f'(x) + O(h) \]

Finite backward difference
Numerical Derivatives
See also T&V, Appendix A.2

Taylor Series expansion

\[
f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + O(h^4)
\]

\[
f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{3!}h^3 f'''(x) + O(h^4)
\]

\[
f(x+h) - f(x-h) = 2hf'(x) + \frac{2}{3!}h^3 f'''(x) + O(h^4)
\]

\[
\frac{f(x+h) - f(x-h)}{2h} = f'(x) + O(h^2)
\]

Finite central difference
Numerical Derivatives
See also T&V, Appendix A.2

Finite forward difference
\[
\frac{f(x + h) - f(x)}{h} = f'(x) + O(h)
\]

Finite backward difference
\[
\frac{f(x) - f(x - h)}{h} = f'(x) + O(h)
\]

Finite central difference
\[
\frac{f(x + h) - f(x - h)}{2h} = f'(x) + O(h^2)
\]

More accurate
Harris Detector: Mathematics

Change of intensity for the shift \([u, v]\):

\[
E(u, v) = \sum_{x,y} w(x, y) \left[ I(x+u, y+v) - I(x, y) \right]^2
\]

Window function  
Shifted intensity  
Intensity

Window function \(w(x, y) = \)

1 in window, 0 outside  
Gaussian
Taylor Series for 2D Functions

\[ f(x+u, y+v) = f(x,y) + uf_x(x,y) + vf_y(x,y) + \]

**First partial derivatives**

\[ \frac{1}{2!} \left[ u^2 f_{xx}(x,y) + uv f_{xy}(x,y) + v^2 f_{yy}(x,y) \right] + \]

**Second partial derivatives**

\[ \frac{1}{3!} \left[ u^3 f_{xxx}(x,y) + u^2 v f_{xxy}(x,y) + uv^2 f_{xyy}(x,y) + v^3 f_{yyy}(x,y) \right] + \]

**Third partial derivatives**

\[ + \ldots \text{ (Higher order terms)} \]

**First order approx**

\[ f(x+u, y+v) \approx f(x,y) + uf_x(x,y) + vf_y(x,y) \]
Lucas-Kanade Derivation

\[
E(u, v) = \sum [I(x+u, y+v) - T(x, y)]^2 \\
\approx \sum [I(x, y) + uI_x(x, y) + vI_y(x, y) - T(x, y)]^2 \\
= \sum [uI_x(x, y) + vI_y(x, y) + D(x, y)]^2 \\
\]

First order approx

Take partial derivs and set to zero

\[
\frac{\delta E}{du} = \sum [uI_x(x, y) + vI_y(x, y) + D(x, y)]I_x(x, y) = 0 \\
\frac{\delta E}{dv} = \sum [uI_x(x, y) + vI_y(x, y) + D(x, y)]I_y(x, y) = 0 \\
\]

Form matrix equation

\[
\sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \sum \begin{bmatrix} I_xD \\ I_yD \end{bmatrix} \\
\]

solve via least-squares
Lucas-Kanade Tracking
Lucas Kanade Tracking

Traditional Lucas-Kanade is typically run on small, corner-like features (e.g. 5x5) to compute optic flow.

Observation: There’s no reason we can’t use the same approach on a larger window around the object being tracked.

Well, actually there is...
Lucas Kanade Tracking

Assumption of constant flow (pure translation) for all pixels in a larger window is unreasonable.

However, we can easily generalize Lucas-Kanade approach to other 2D parametric motion models (like affine or projective)
Warping Review

FIGURE 1. Basic set of 2D planar transformations

picture from R. Szeliski
Geometric Image Mappings

\[ x' = f(x, y, \{\text{parameters}\}) \]
\[ y' = g(x, y, \{\text{parameters}\}) \]
Linear Transformations
(Can be written as matrices)

\[
\begin{pmatrix}
  x' \\
  y' \\
  1
\end{pmatrix}
= 
M(\text{params})
\begin{pmatrix}
  x \\
  y \\
  1
\end{pmatrix}
\]
Translation

Transform

\[ x' = x + t_x \]
\[ y' = y + t_y \]

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
= \begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

equations

matrix form
Translation

Transform

\[ x' = x + t_x \]
\[ y' = y + t_y \]

Equations

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Matrix form
Scale

Transform

Equations

Matrix form

\[ x' = s x_i \]
\[ y' = s y_i \]
Rotation

\[ x' = x \cos \theta - y \sin \theta \]
\[ y' = x \sin \theta + y \cos \theta \]

Equations

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
= \begin{bmatrix}
  \cos \theta & -\sin \theta & 0 \\
  \sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Matrix form
Euclidean (Rigid)

\[ x' = x \cos \theta - y \sin \theta + t_x \]
\[ y' = x \sin \theta + y \cos \theta + t_y \]

Equations

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
= 
\begin{bmatrix}
  \cos \theta & -\sin \theta & t_x \\
  \sin \theta & \cos \theta & t_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Matrix form
Partitioned Matrices

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta & t_x \\
\sin \theta & \cos \theta & t_y \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

\[
\begin{bmatrix}
p' \\
1
\end{bmatrix} = \begin{bmatrix}
R & t \\
0 & 1
\end{bmatrix} \begin{bmatrix}
p \\
1
\end{bmatrix}
\]

\[
p' = Rp + t
\]

matrix form

equation form
Similarity (scaled Euclidean)

\[ p' = sR p + t \]

Equations

\[
\begin{bmatrix}
  p' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  sR & t \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  p \\
  1
\end{bmatrix}
\]

Matrix form
Affine matrix form

\[ p' = Ap + b \]

\[
\begin{bmatrix}
    p' \\
    1
\end{bmatrix} =
\begin{bmatrix}
    A & b \\
    0 & 1
\end{bmatrix}
\begin{bmatrix}
    p \\
    1
\end{bmatrix}
\]
Projective

\[ p' = \frac{Ap + b}{c^T p + 1} \]

\[
\begin{bmatrix}
  p' \\
  1
\end{bmatrix} \sim
\begin{bmatrix}
  A & b \\
  c^T & 1
\end{bmatrix}
\begin{bmatrix}
  p \\
  1
\end{bmatrix}
\]
Summary of 2D Transformations

- Rotation
- Translation
- Scale
- Aspect Ratio
- Skew
- Perspective Warp
Summary of 2D Transformations

Euclidean

- rotation
- translation
- scale
- aspect ratio
- skew
- perspective warp
Summary of 2D Transformations

Similarity

- Rotation
- Translation
- Scale
- Aspect ratio
- Skew
- Perspective warp
Summary of 2D Transformations

Affine

- rotation
- translation
- scale
- aspect ratio
- skew
- perspective warp
Summary of 2D Transformations

Projective

rotation  translation  scale

aspect ratio  perspective warp

skew
## Summary of 2D Transformations

<table>
<thead>
<tr>
<th>Name</th>
<th>Matrix</th>
<th># D.O.F.</th>
<th>Preserves:</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>translation</td>
<td>$\begin{bmatrix} I &amp; t \end{bmatrix}_{2\times3}$</td>
<td>2</td>
<td>orientation + ⋯</td>
<td></td>
</tr>
<tr>
<td>rigid (Euclidean)</td>
<td>$\begin{bmatrix} R &amp; t \end{bmatrix}_{2\times3}$</td>
<td>3</td>
<td>lengths + ⋯</td>
<td></td>
</tr>
<tr>
<td>similarity</td>
<td>$\begin{bmatrix} sR &amp; t \end{bmatrix}_{2\times3}$</td>
<td>4</td>
<td>angles + ⋯</td>
<td></td>
</tr>
<tr>
<td>affine</td>
<td>$\begin{bmatrix} A \end{bmatrix}_{2\times3}$</td>
<td>6</td>
<td>parallelism + ⋯</td>
<td></td>
</tr>
<tr>
<td>projective</td>
<td>$\begin{bmatrix} H \end{bmatrix}_{3\times3}$</td>
<td>8</td>
<td>straight lines</td>
<td></td>
</tr>
</tbody>
</table>
to be continued...