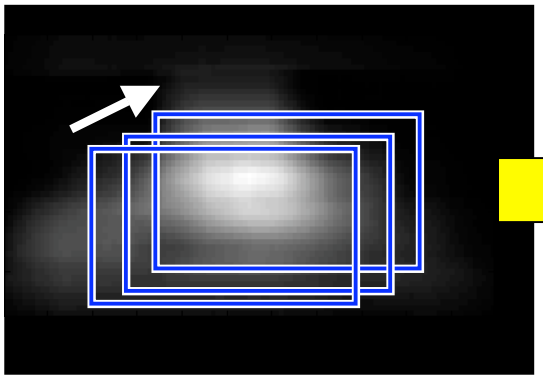
Robert Collins
CSE598G

Intro to Template Matching and the Lucas-Kanade Method

Appearance-Based Tracking

current frame +
previous location

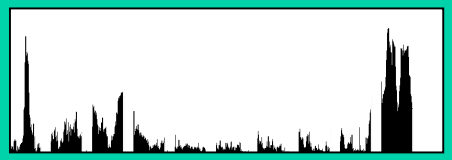
likelihood over
object location



current location

appearance model
(e.g. image template, or

color; intensity; edge histograms)



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

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

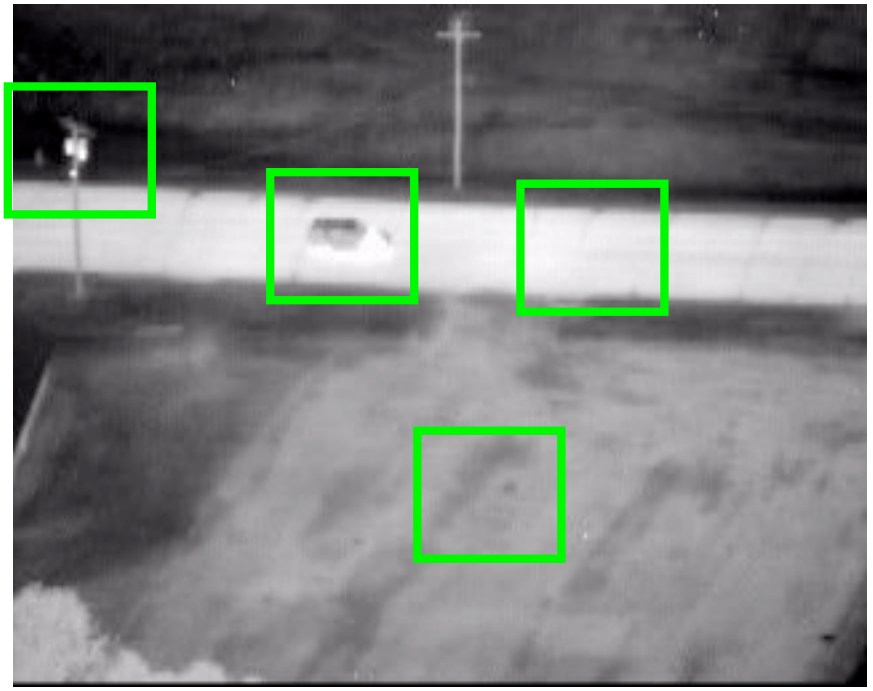
The last two are very restrictive. We will relax them later on.

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

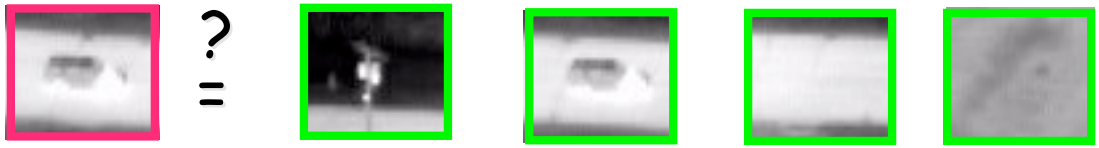
Elements to be matched are image patches of fixed size



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

Correlation-based Algorithms

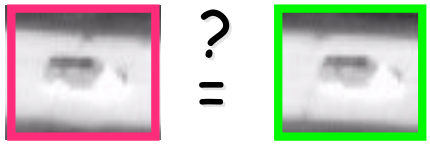
Task: what is the corresponding patch in a second image?



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:



f g

Some possible measures:

$$\stackrel{?}{=} \max_{[i,j]\in R} |f(i,j) - g(i,j)|$$

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

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

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

Most popular

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.

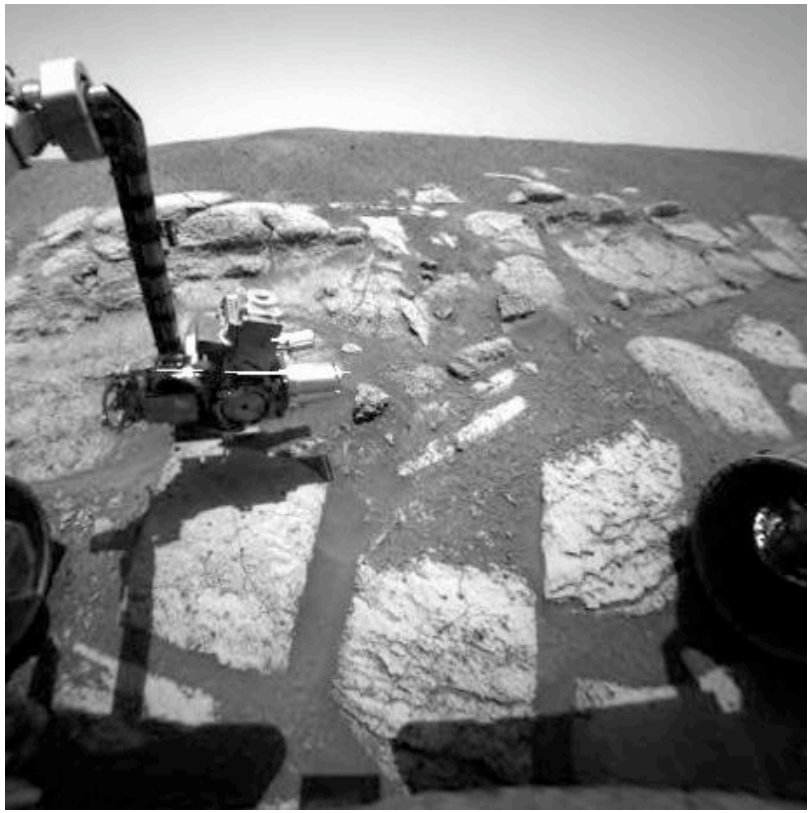


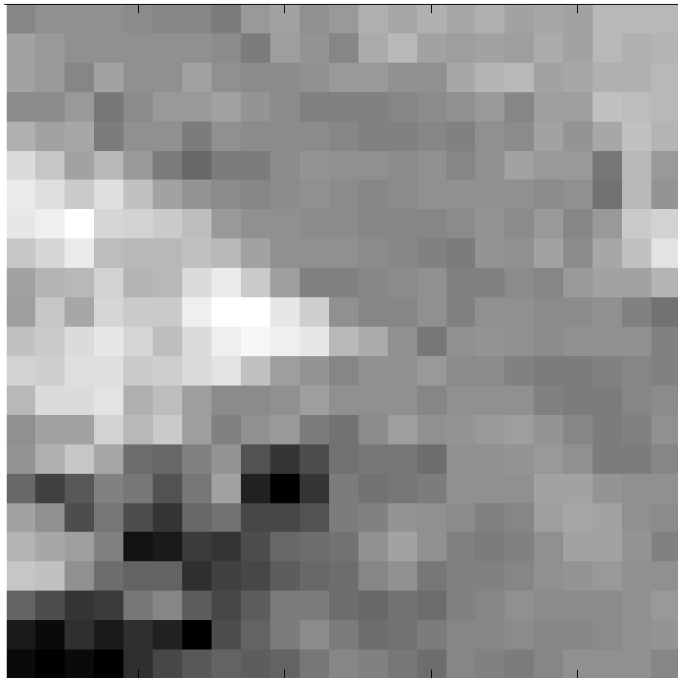
Image 1

Image 2

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

Example

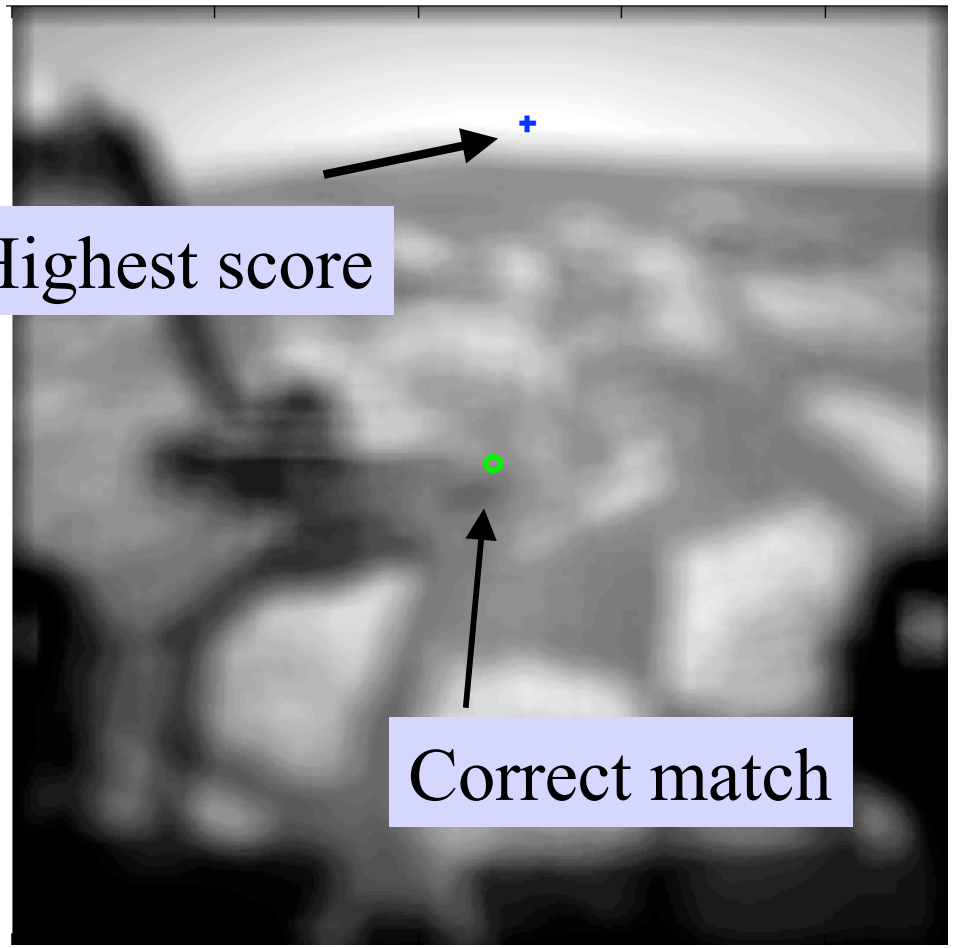
Image 1



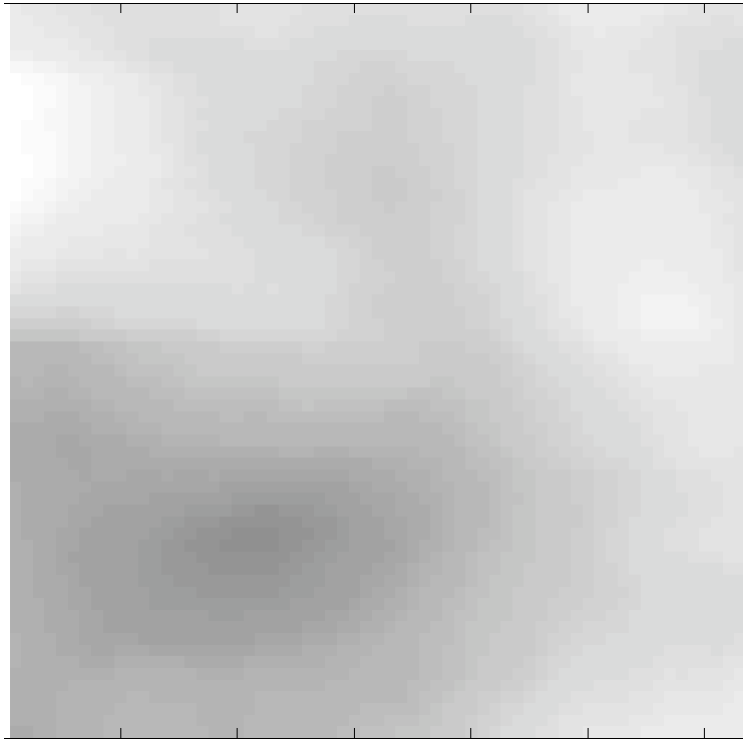
Template
(image patch)

Example: Cross-correlation

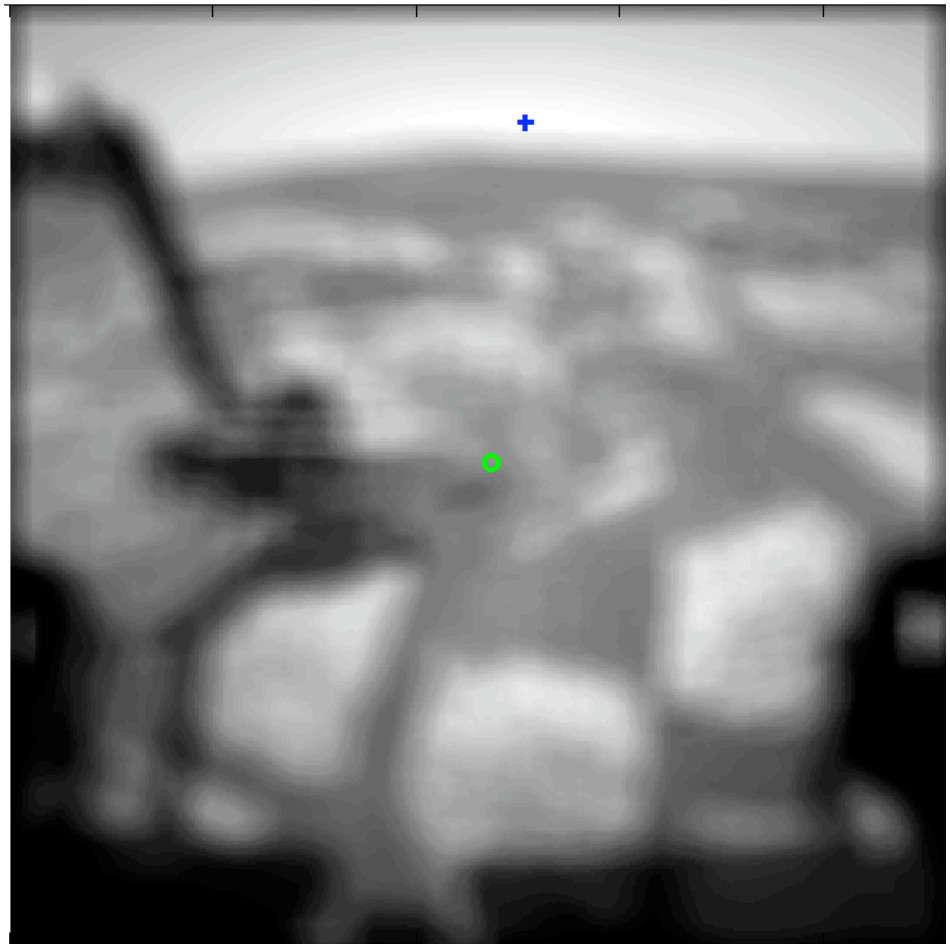
score = imfilter(image2,tmpl)



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:

a	b	c
d	e	f
g	h	i

ⓧ

v	v	v
v	v	v
v	v	v

Result: v*(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.

a	b	c
d	e	f
g	h	i

⊗

2v	2v	2v
2v	2v	2v
2v	2v	2v

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

> v*(a+b+c+d+e+f+g+h+i)

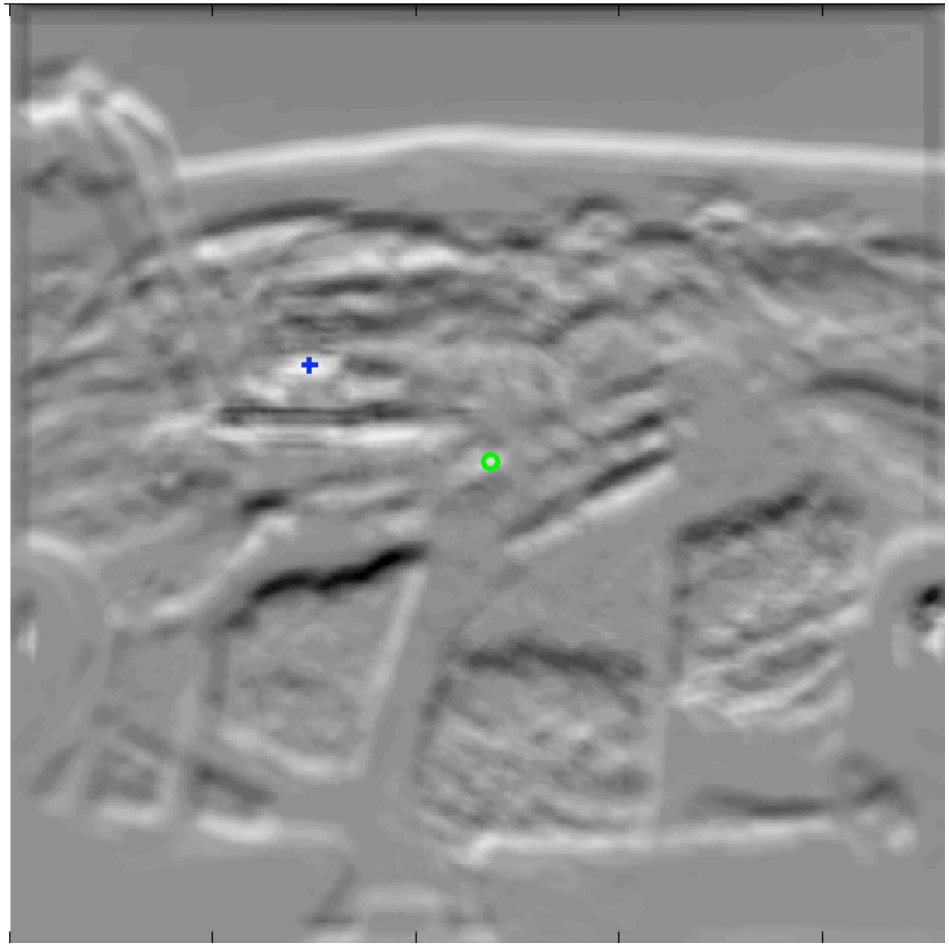
Larger score, regardless of what the template is!

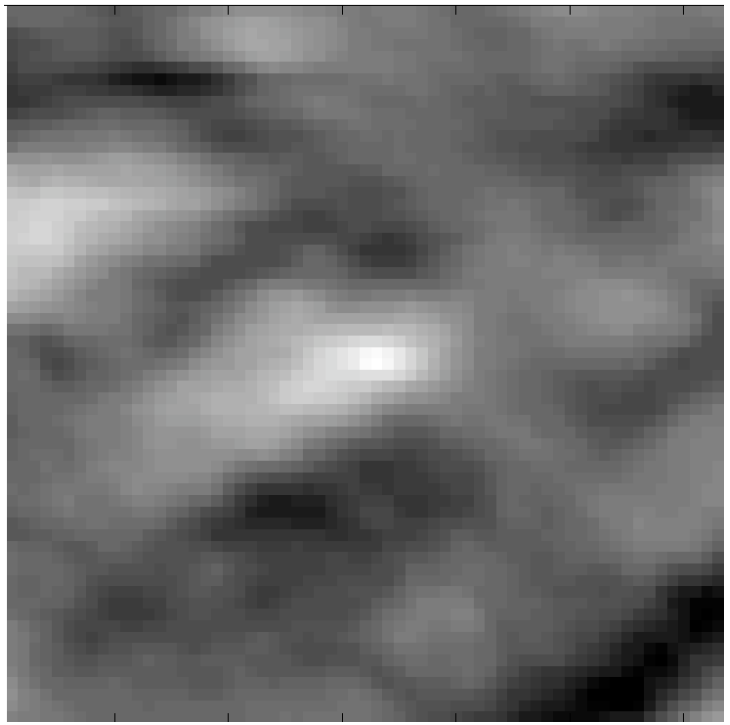
Solution

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

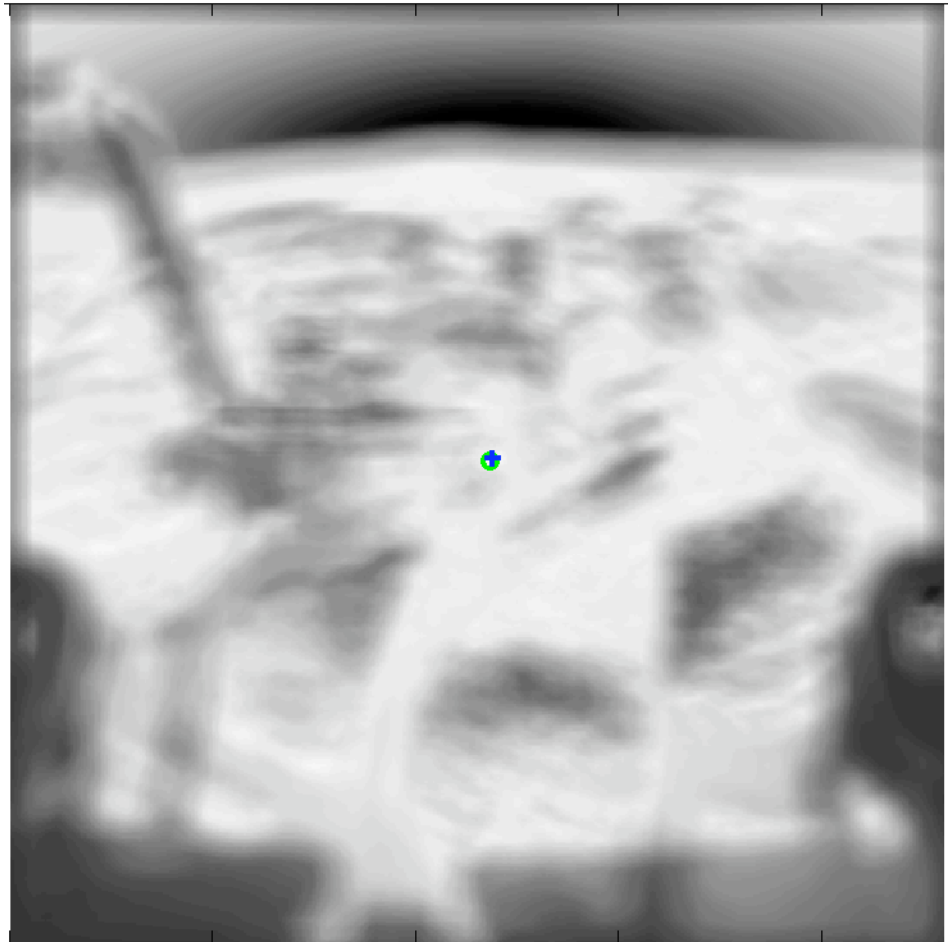
$$SSD = \sum_{[i,j] \in R} (f - g)^2$$

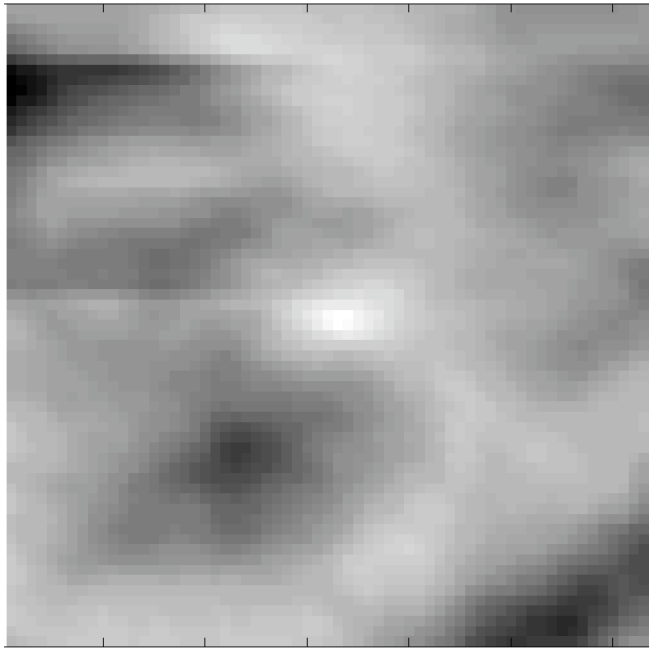
$$= \sum_{[i,j] \in R} f^2 + \sum_{[i,j] \in R} g^2 - 2 \sum_{[i,j] \in R} fg$$

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

Correlation!

SSD





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

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)

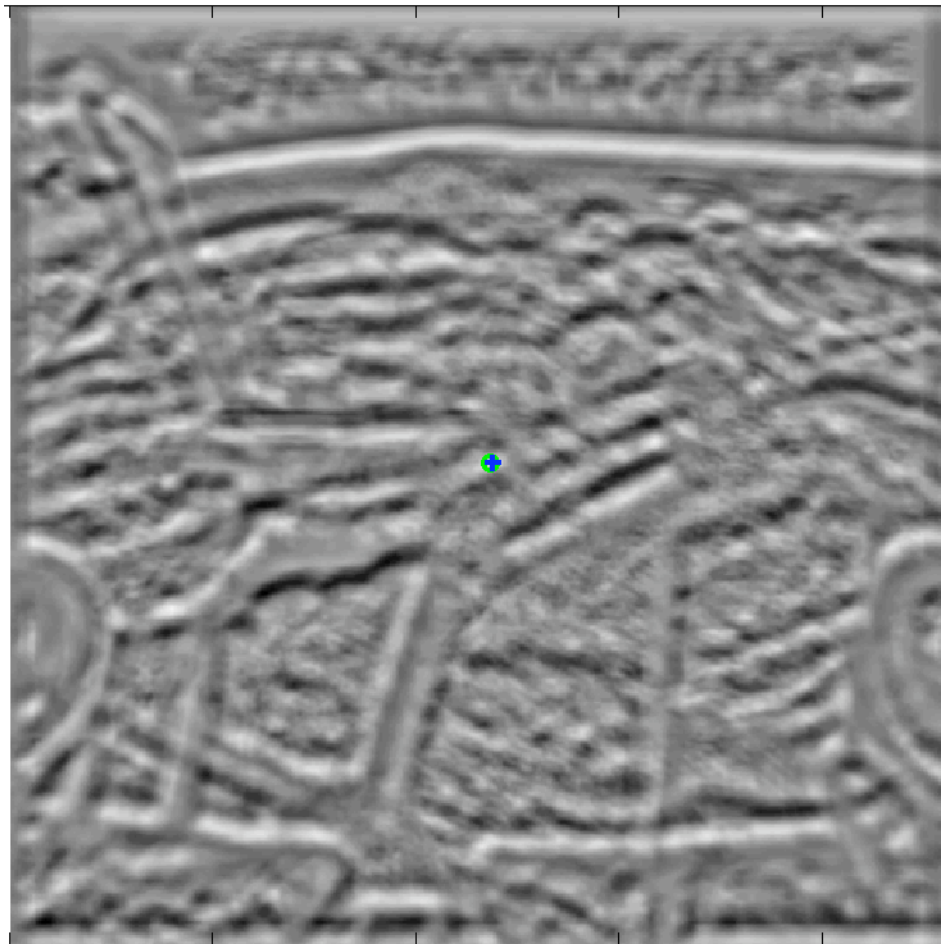
• The second approach is to use a normalized matching function that is invariant to intensity changes. This is the one we will be discussing now.

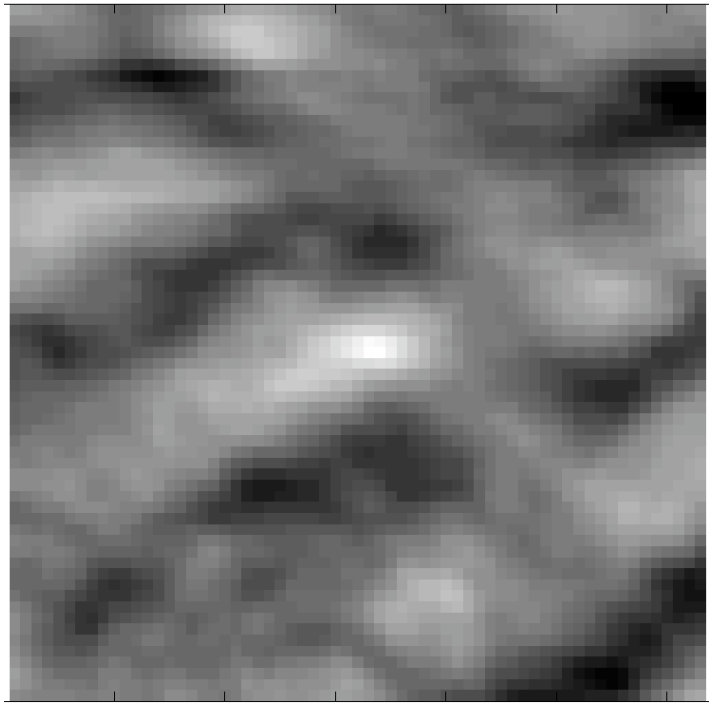
Intensity Normalization

- 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.

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

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 = gain * f + offset

Consider the normalized patches:

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

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 Sample Code

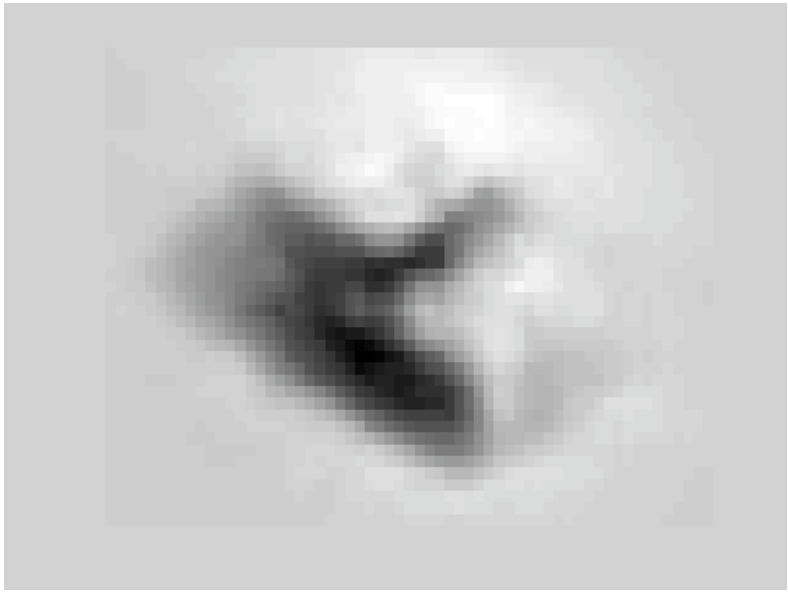
Properties:

- 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)

- I'll put code on the course web site

Normalized Correlation, Fixed Template

template

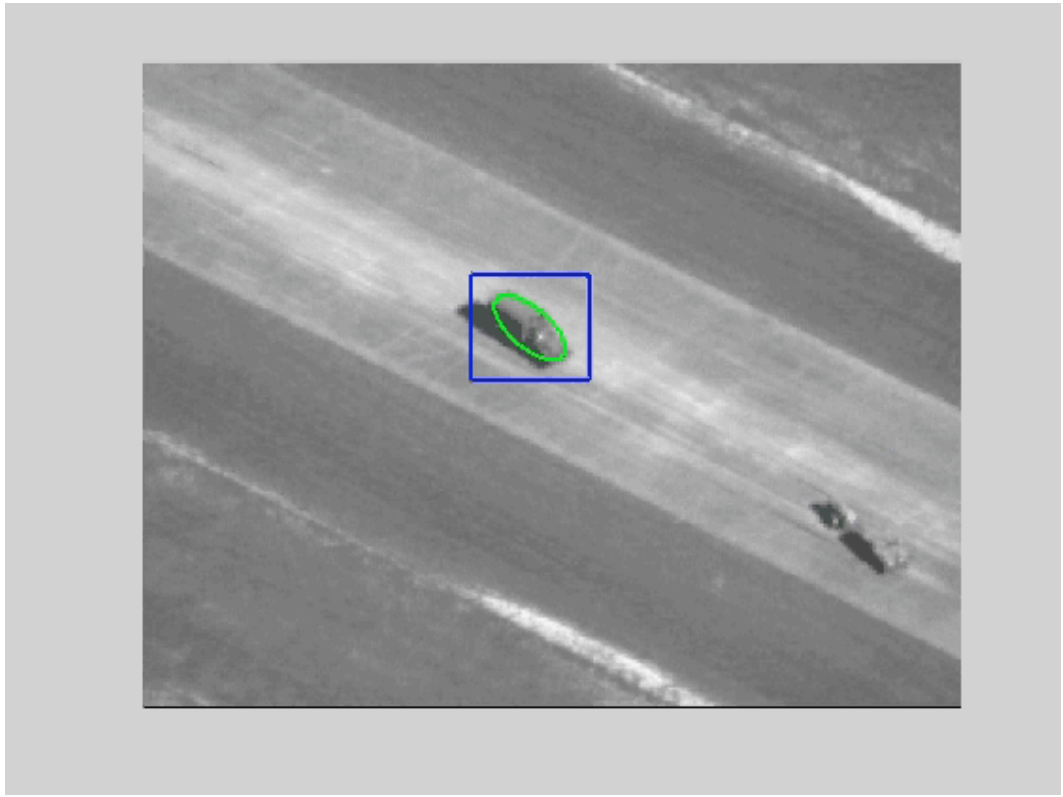


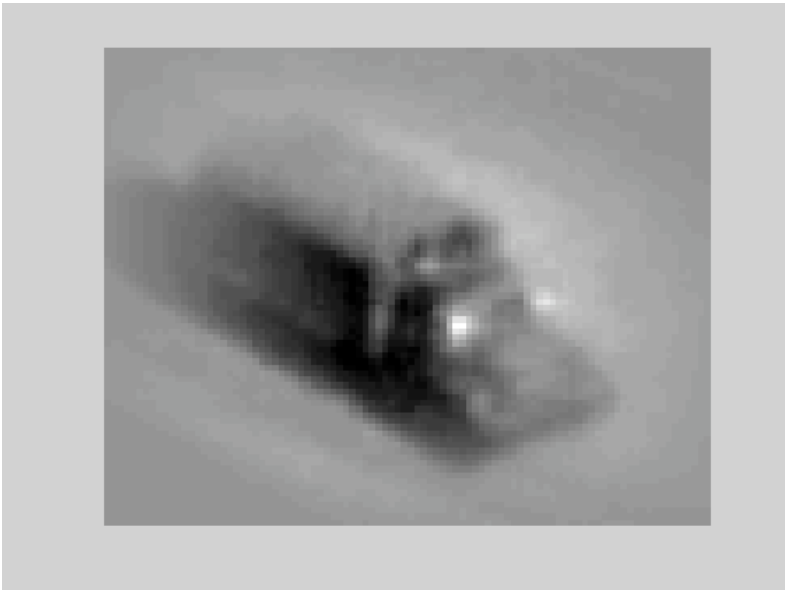
Failure mode: distraction by background clutter

Robert Collins
CSE598G

Normalized Correlation, Fixed Template

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?

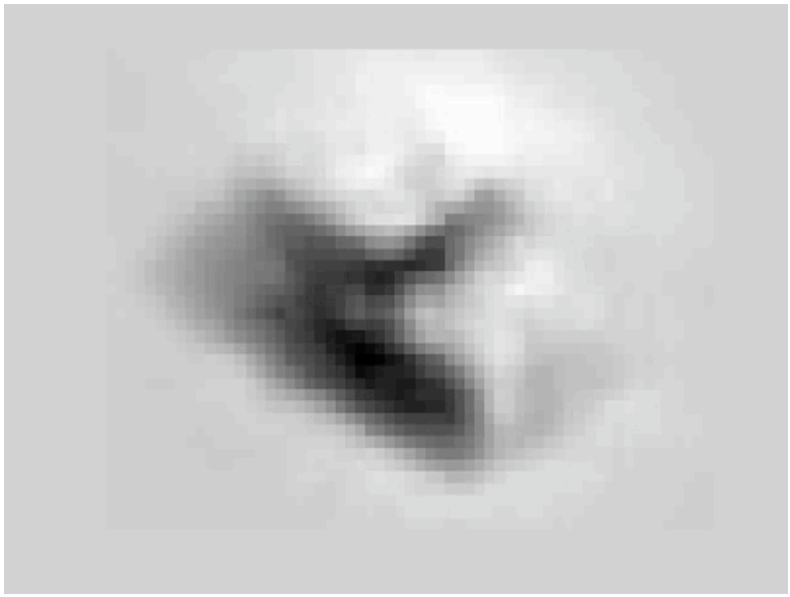
Template Drift

- If your estimate of template location is slightly off, you are now looking for a matching position that is similarly off center.
- Over time, this offset error builds up until the template starts to “slide” off the object.

- The problem of drift is a major issue with methods that adapt to changing object appearance.

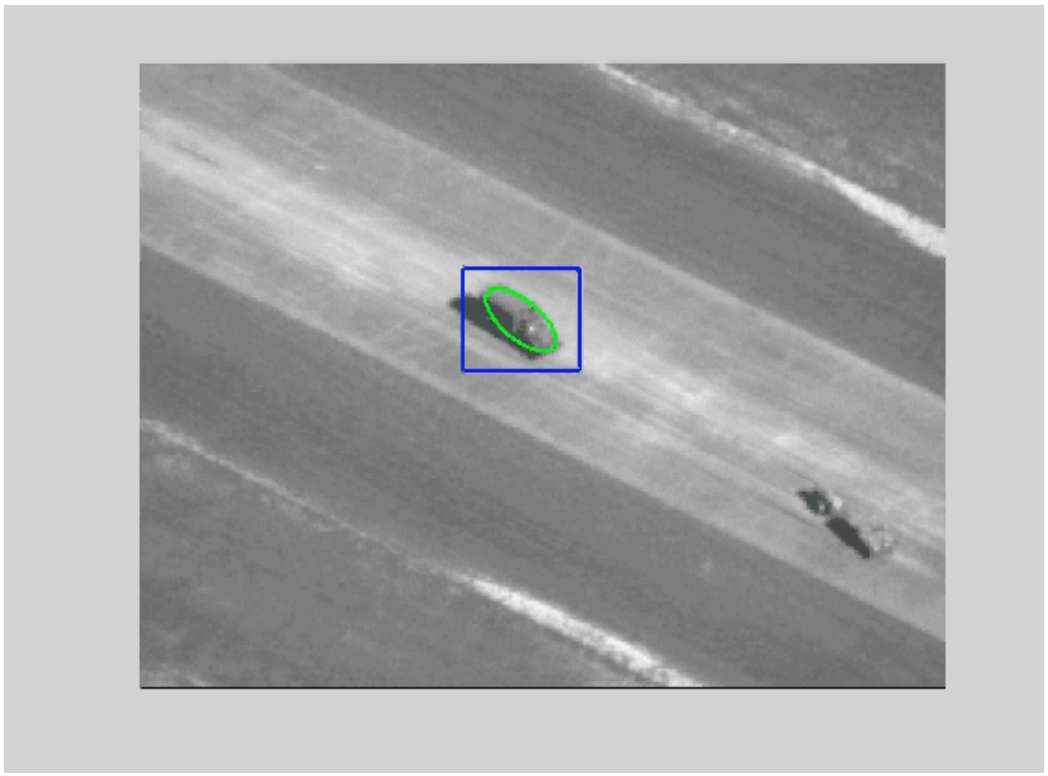
Normalized Correlation, Adaptive Template

template

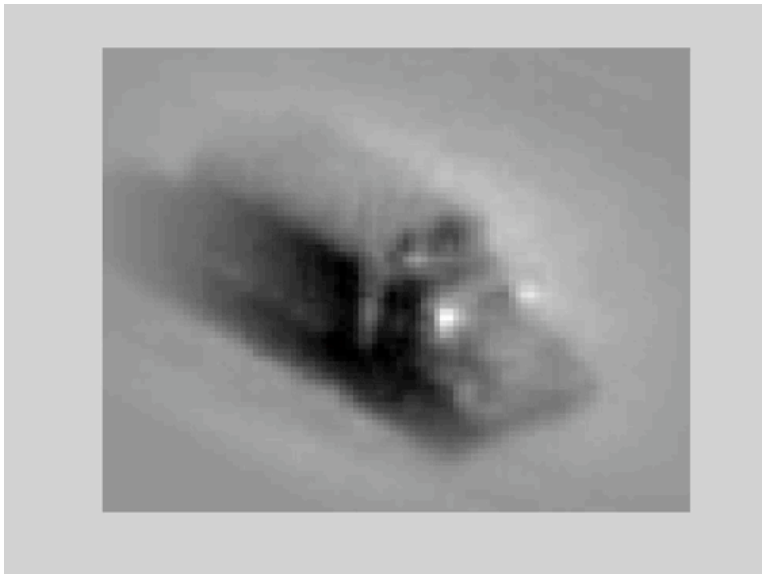


Here our results are no better than before.

Normalized Correlation, Adaptive Template



template



Here the result is even worse than before!

Robert Collins
CSE598G

Tracking via Gradient Descent

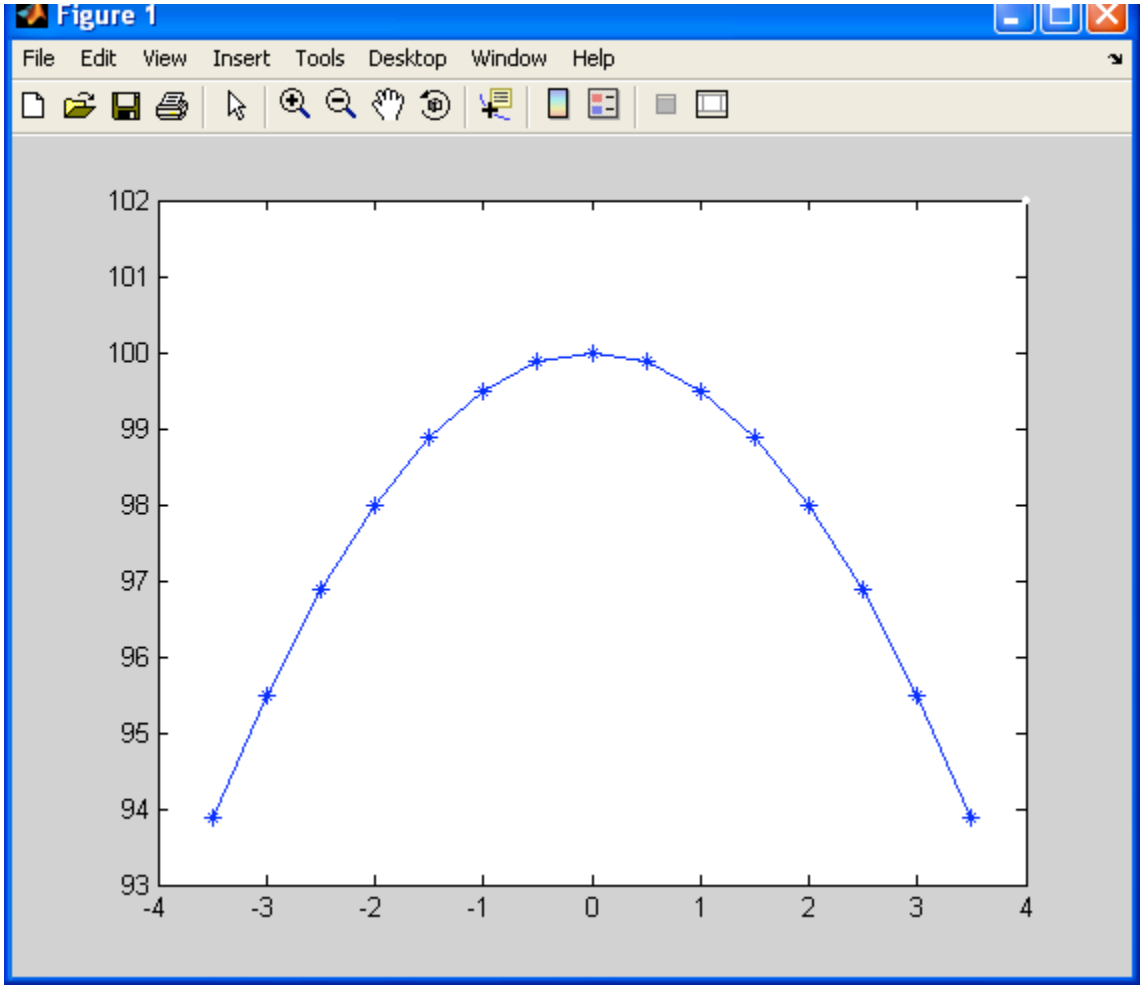
Motivation

- 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 * x^2



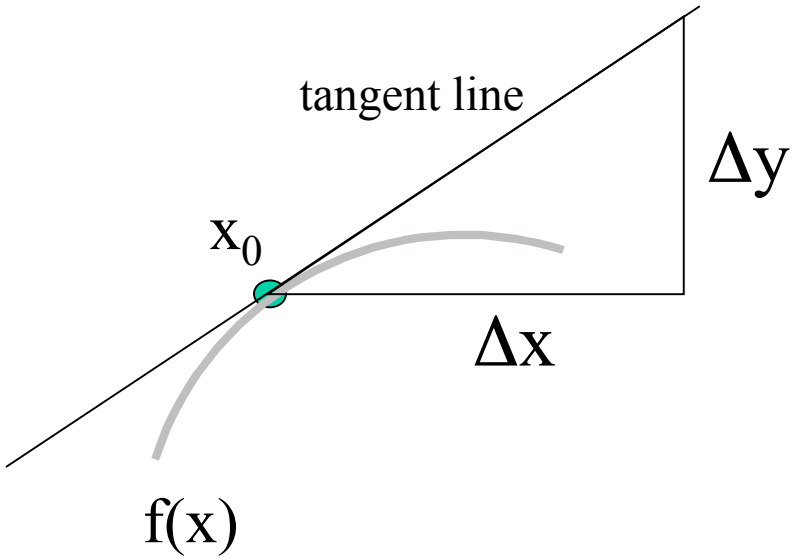
Math Example : 1D Gradient

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

Gradient is df(x)/dx = - 2 * 0.5 * x = - x

Geometric interpretation:

gradient at x_0 is slope of tangent line to curve at point x_0

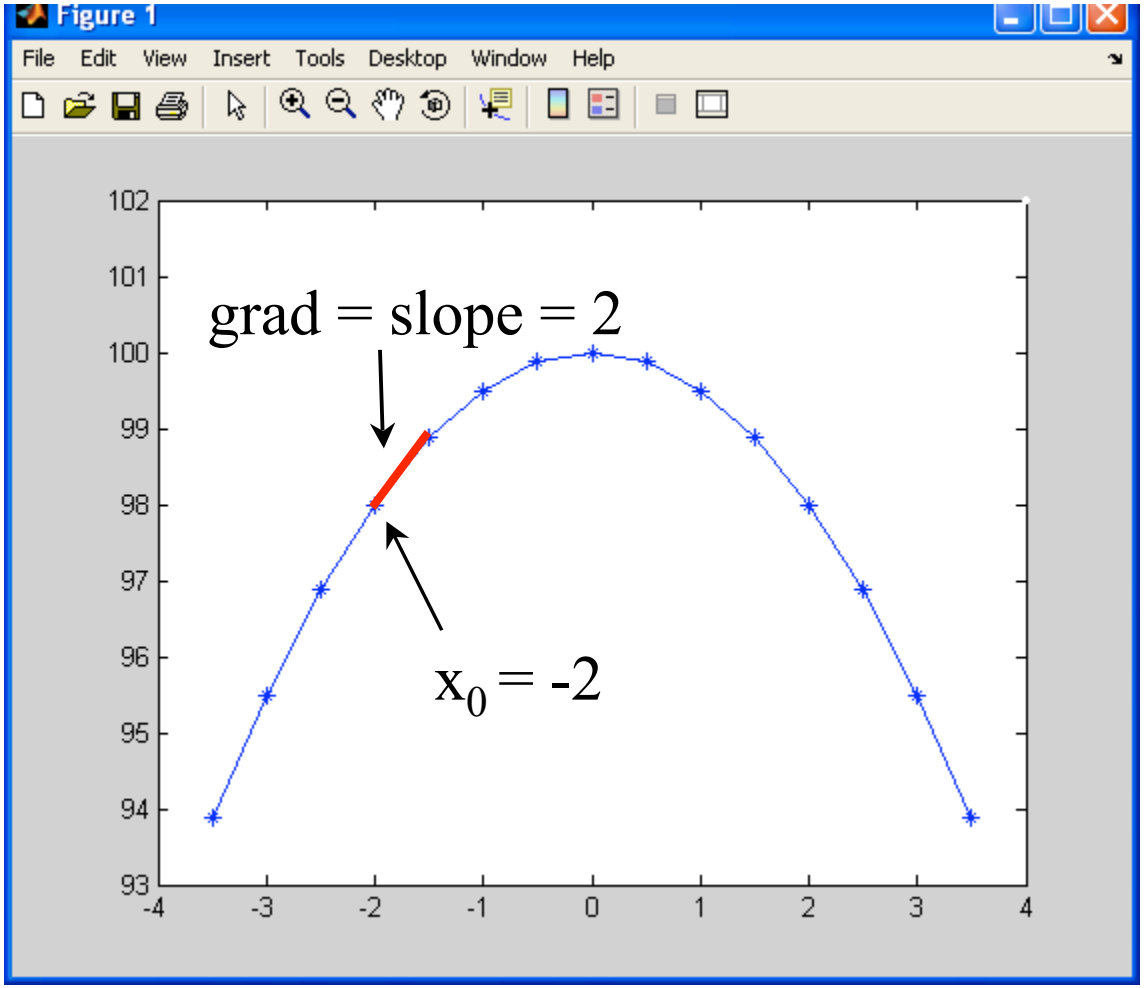


$$\text{slope} = \Delta y / \Delta x$$

$$= df(x)/dx \Big|_{x_0}$$

Math Example : 1D Gradient

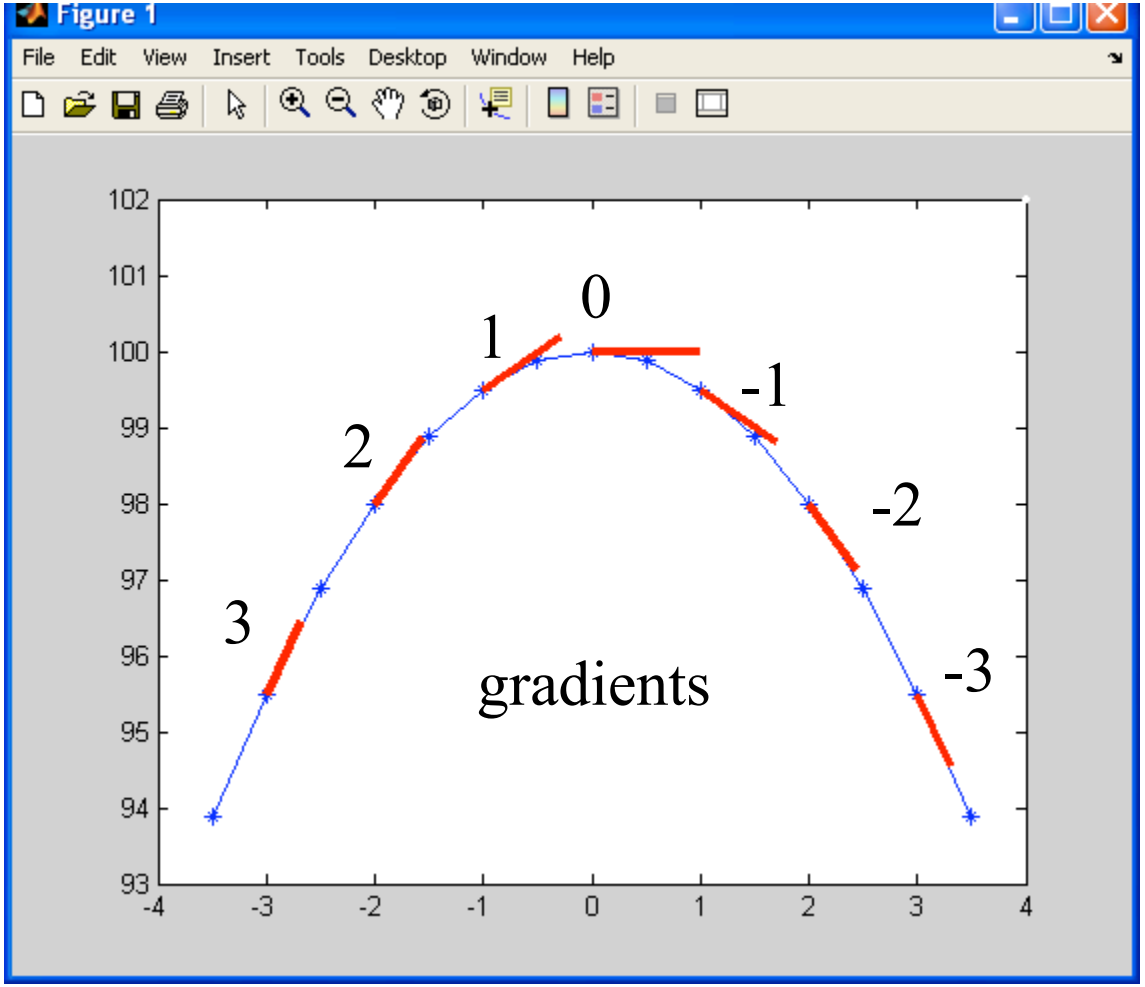
f(x) = 100 - 0.5 * x^2 df(x)/dx = - x



Robert Collins
CSE598G

Math Example : 1D Gradient

$f(x) = 100 - 0.5 * x^2 \qquad df(x)/dx = - x$

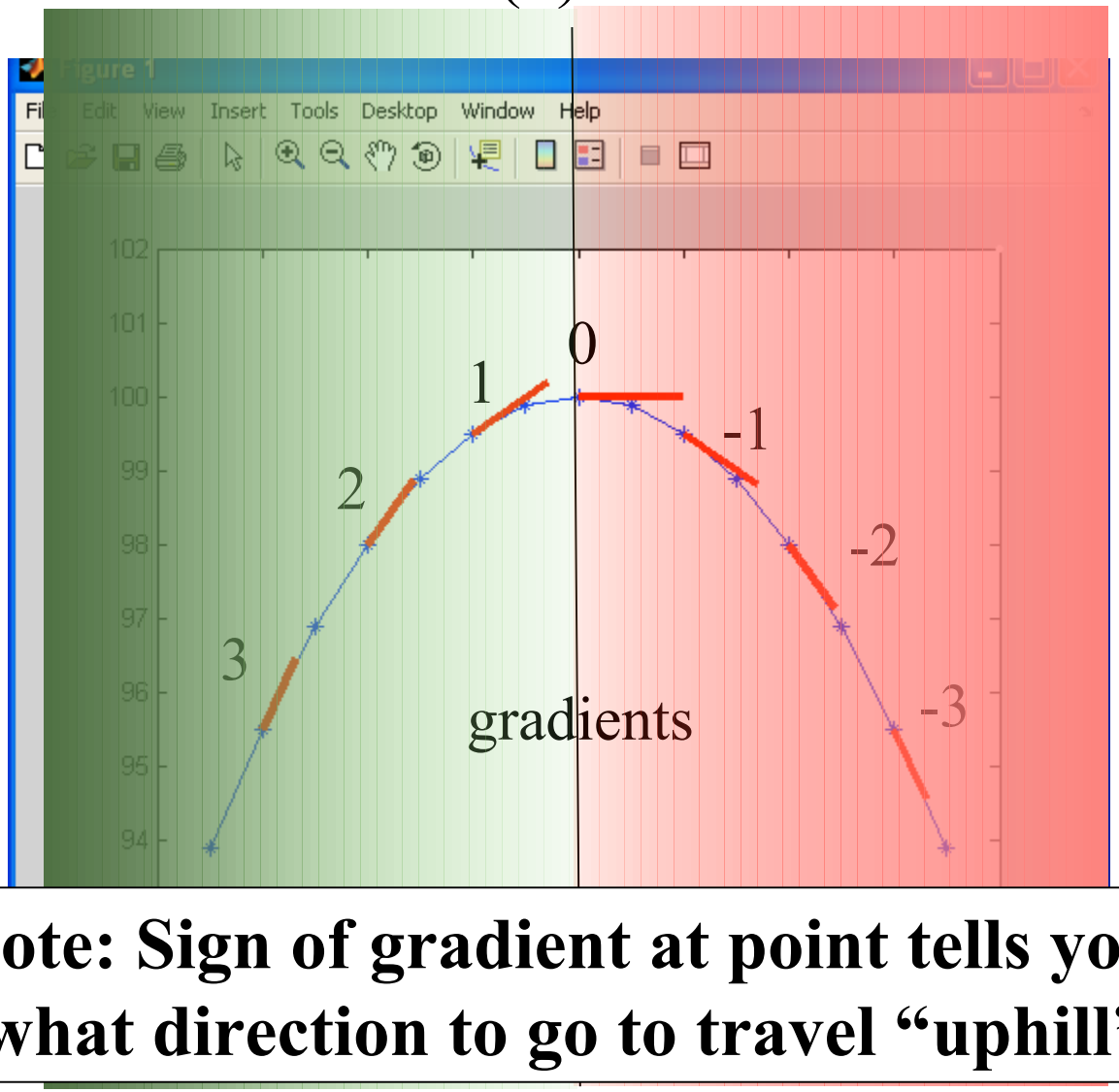


Robert Collins
CSE598G

Math Example : 1D Gradient

f(x) = 100 - 0.5 * x^2 df(x)/dx = - x

Gradients on this side of peak are positive



Gradients on this side of peak are negative

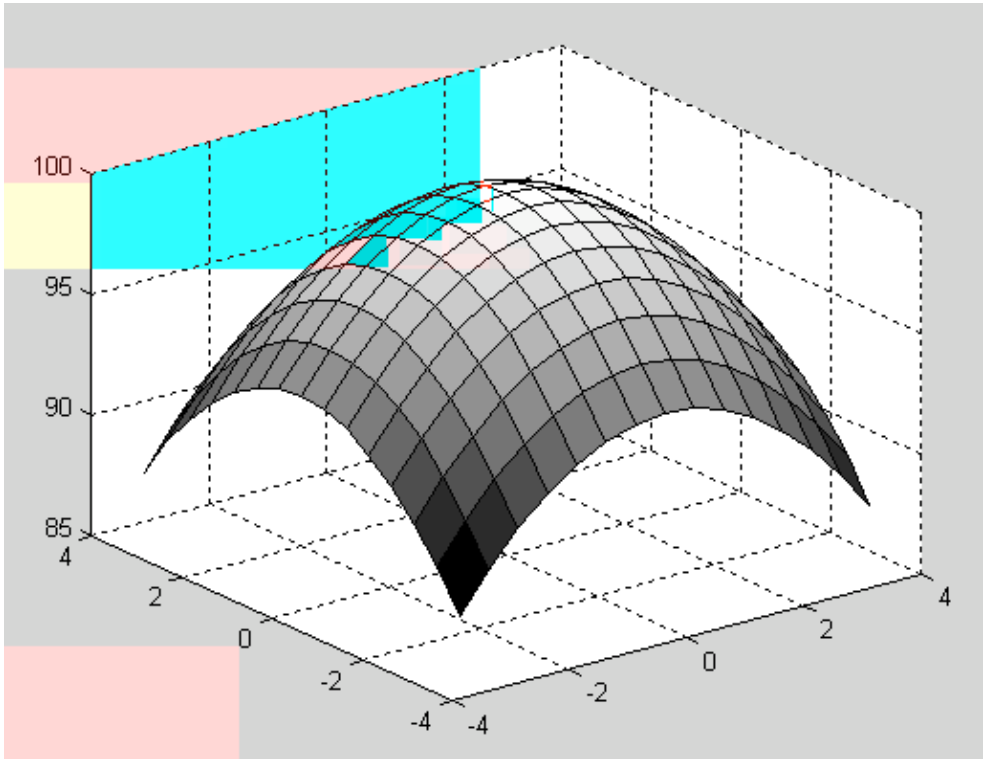
Note: Sign of gradient at point tells you what direction to go to travel "uphill"

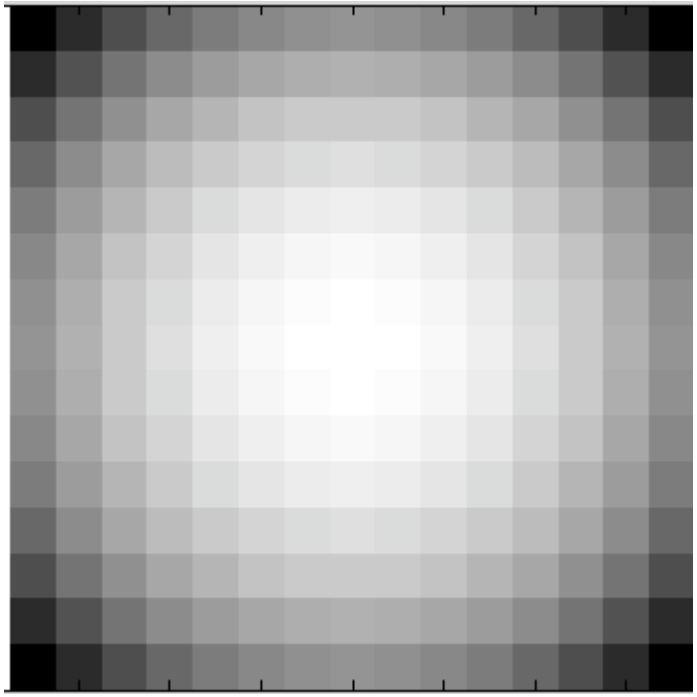
Math Example : 2D Gradient

$$f(x,y) = 100 - 0.5 * x^2 - 0.5 * y^2$$

$$df(x,y)/dx = -x \qquad df(x,y)/dy = -y$$

$$\text{Gradient} = [df(x,y)/dx , \; df(x,y)/dy] = [-x , -y]$$



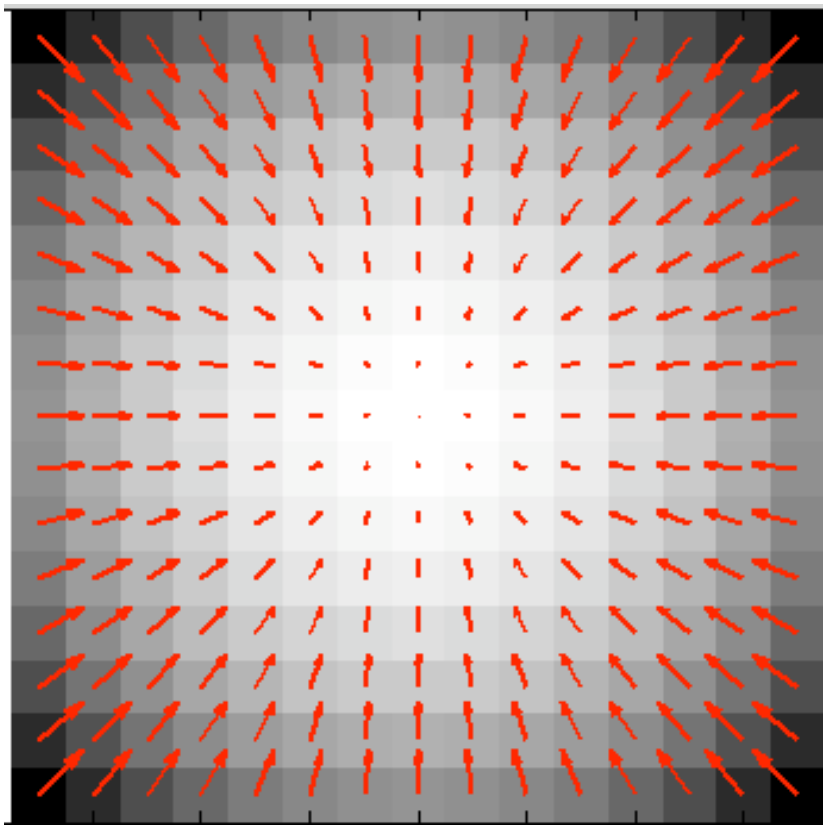


Gradient is vector of partial derivs wrt x and y axes

Math Example : 2D Gradient

$$f(x,y) = 100 - 0.5 * x^2 - 0.5 * y^2$$

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



Plotted as a vector field, the gradient vector at each pixel points "uphill"

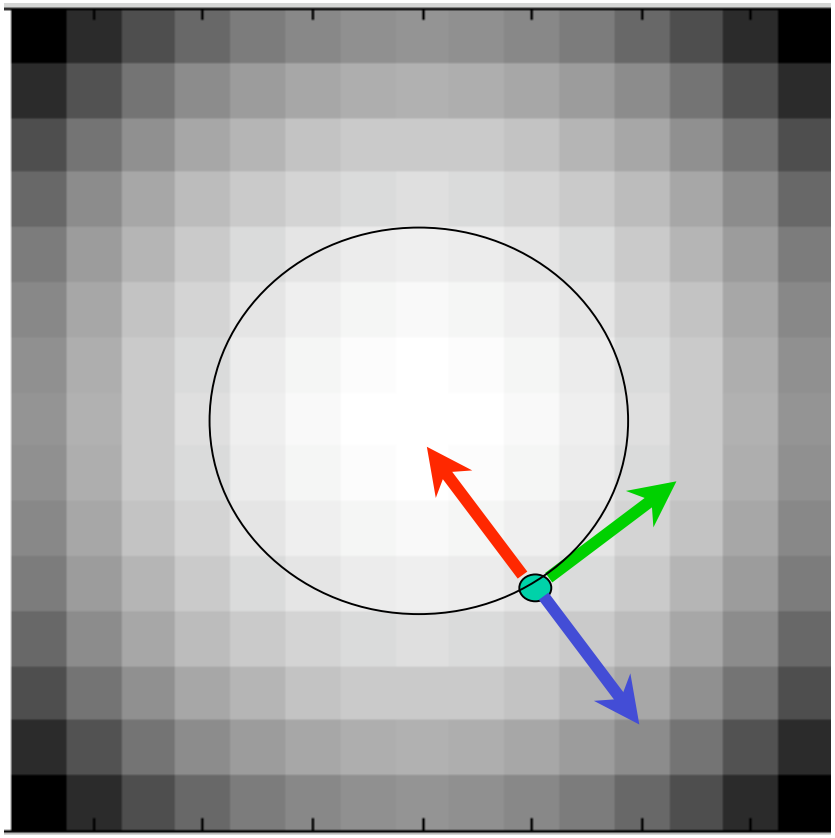
The gradient indicates the direction of steepest ascent.

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

$$f(x,y) = 100 - 0.5 * x^2 - 0.5 * y^2$$

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



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)

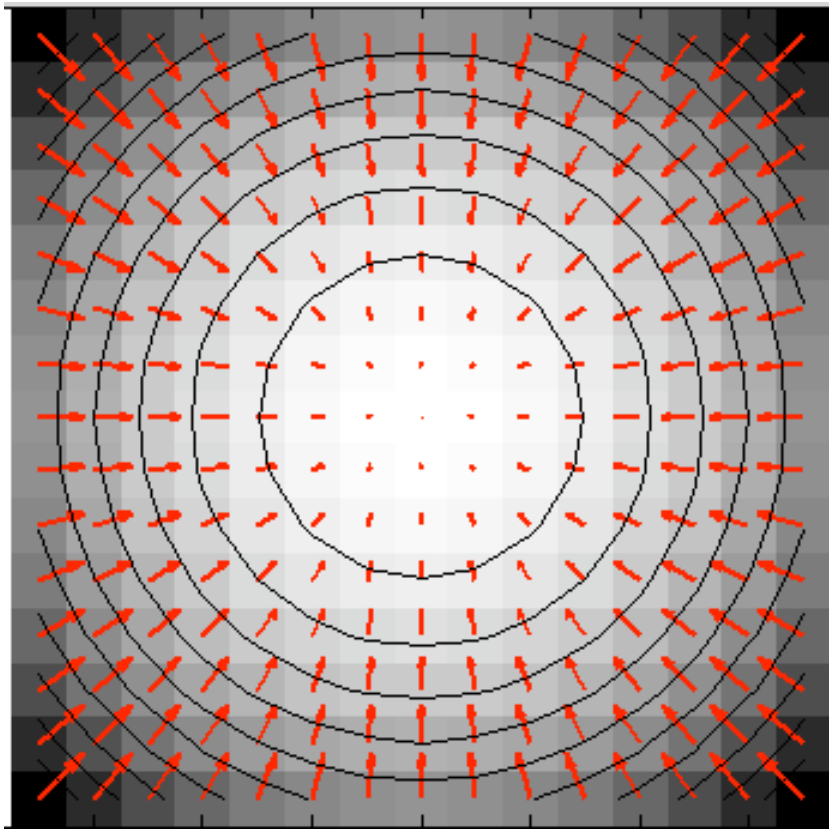
Vector - g points downhill
(direction of steepest descent)

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

$$f(x,y) = 100 - 0.5 * x^2 - 0.5 * y^2$$

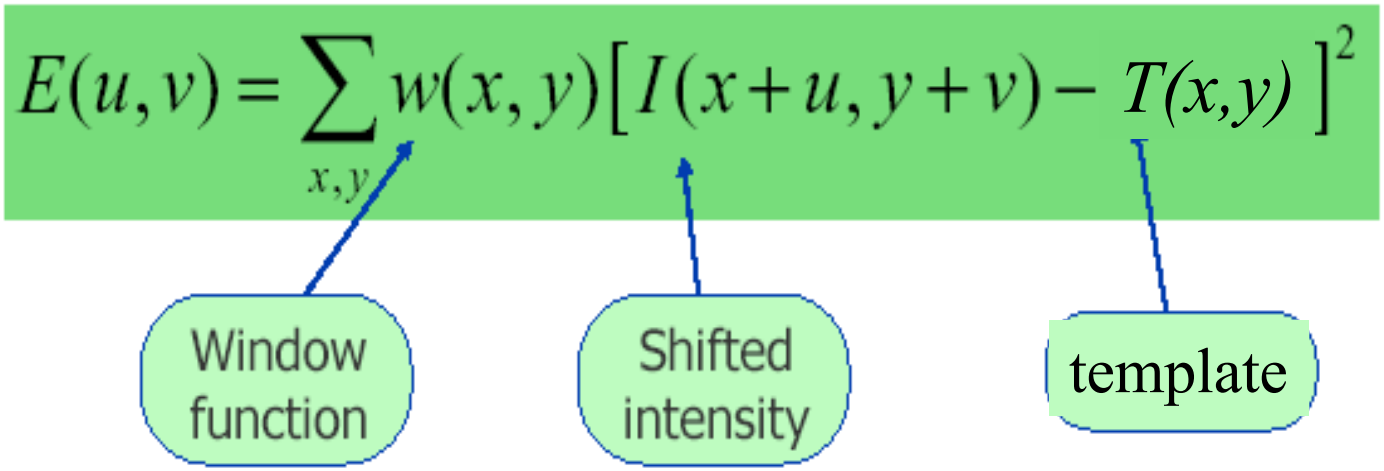
$$\text{Gradient} = [df(x,y)/dx ,\ df(x,y)/dy] = [- x , - y]$$



And so on for all points

What is Relevance of This?

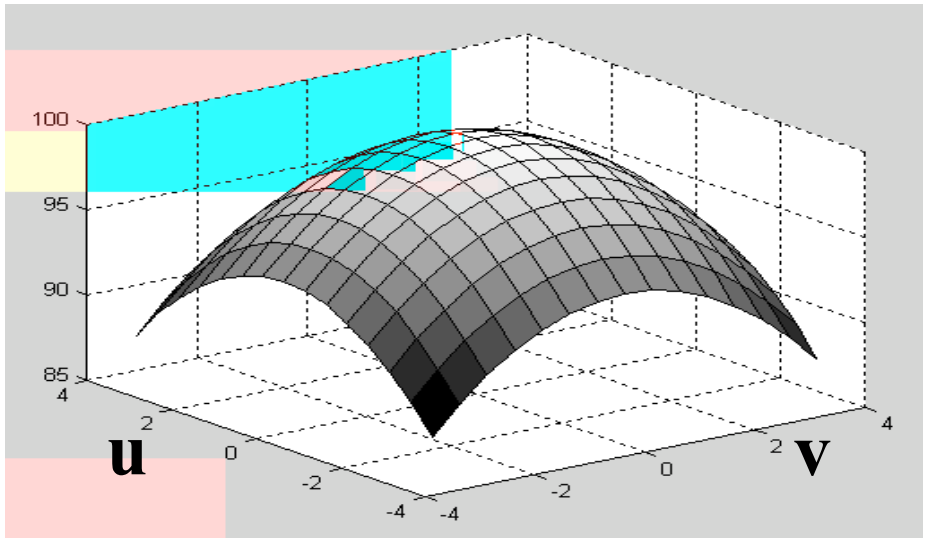
consider the SSD error function



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.

Robert Collins
CSE598G

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

Robert Collins
CSE598G

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

Robert Collins
CSE598G

Numerical Derivatives

See also T&V, Appendix A.2

Taylor Series expansion

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

subtract

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

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

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

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 $[u,v]$:

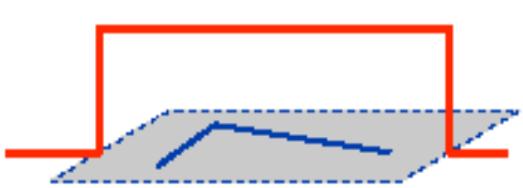
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

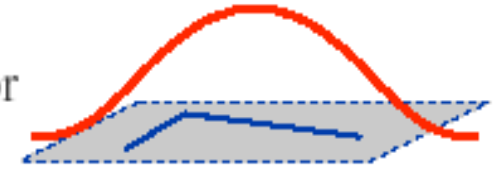
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Robert Collins
CSE598G

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

$+ \ldots$ (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

$$E(u,v) = \sum [I(x+u,y+v) - T(x,y)]^2$$

$$\approx \sum [I(x,y) + uI_x(x,y) + vI_y(x,y) - T(x,y)]^2 \quad \text{First order approx}$$

$$= \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]^2$$

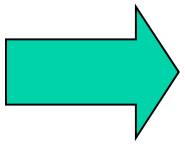
Take partial derivs and set to zero

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

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

Form matrix equation

$$\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\sum \begin{bmatrix} I_x D \\ I_y D \end{bmatrix}$$



solve via least-squares

Robert Collins
CSE598G

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)

Robert Collins
CSE598G

Warping Review

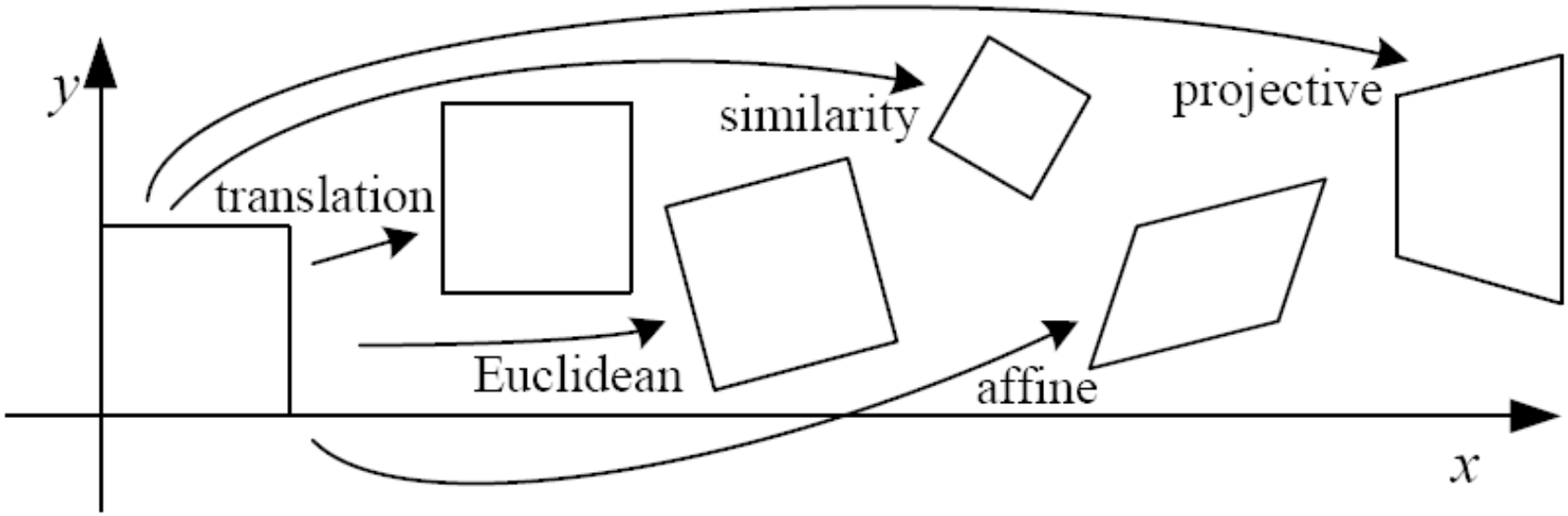
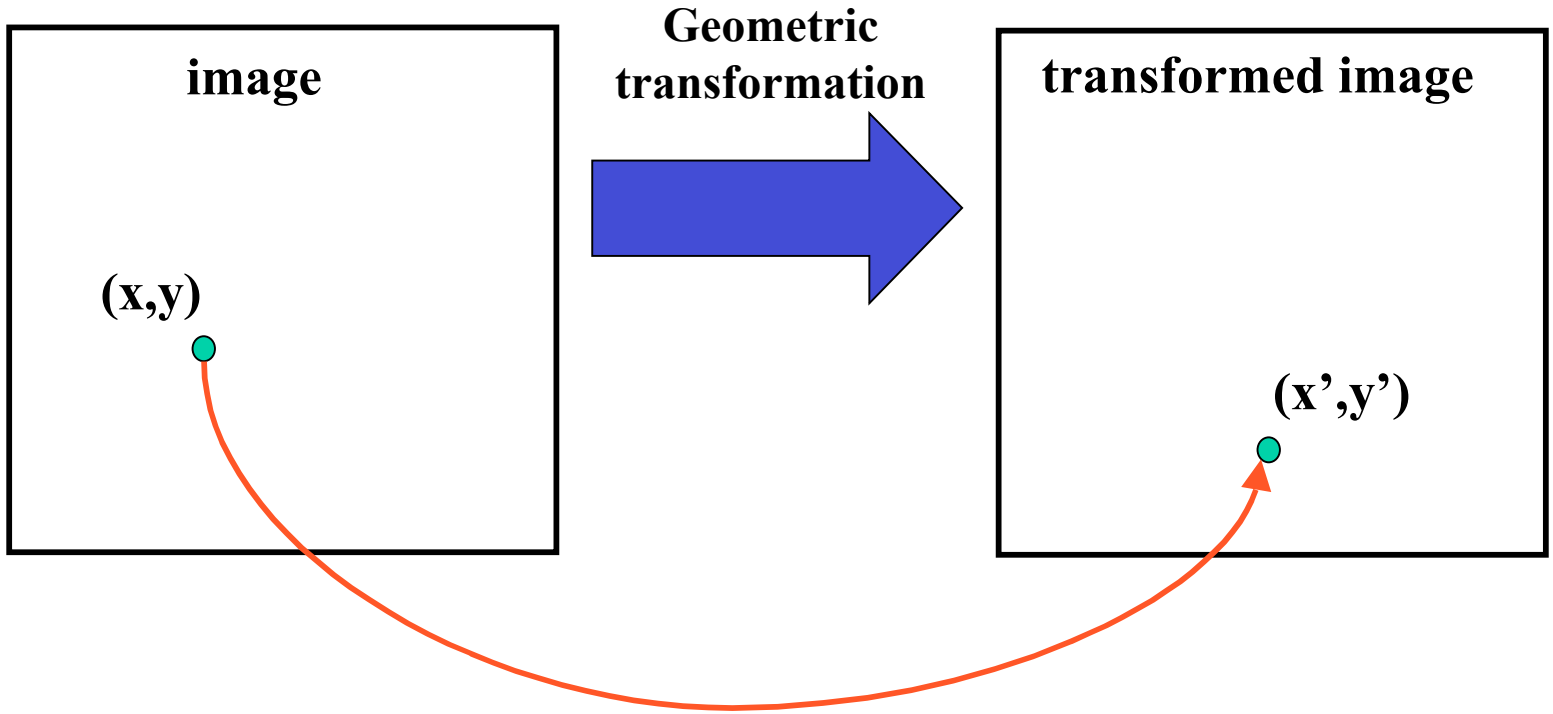


FIGURE 1. Basic set of 2D planar transformations

picture from R.Szeliski

Robert Collins
CSE598G

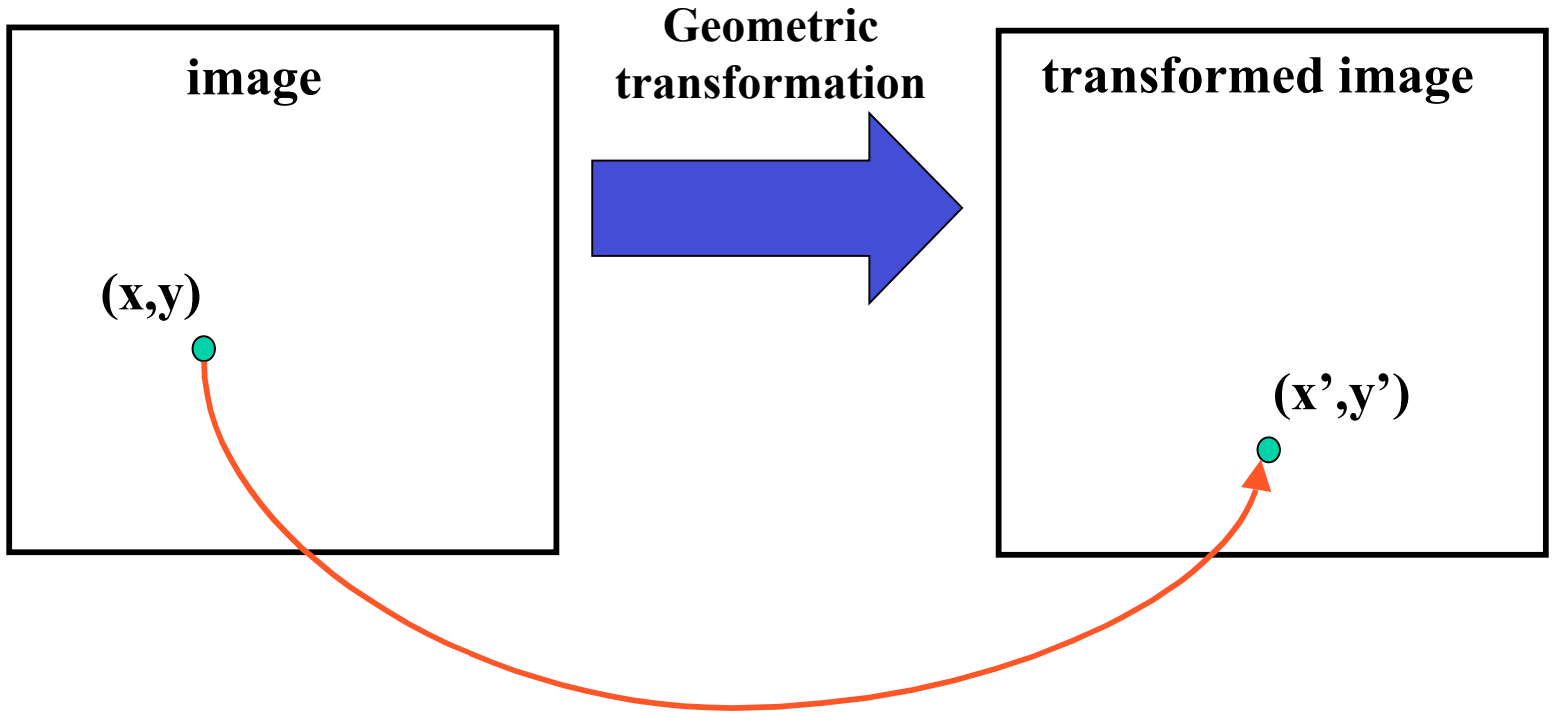
Geometric Image Mappings



$$x' = f(x, y, \{parameters\})$$
$$y' = g(x, y, \{parameters\})$$

Robert Collins
CSE598G

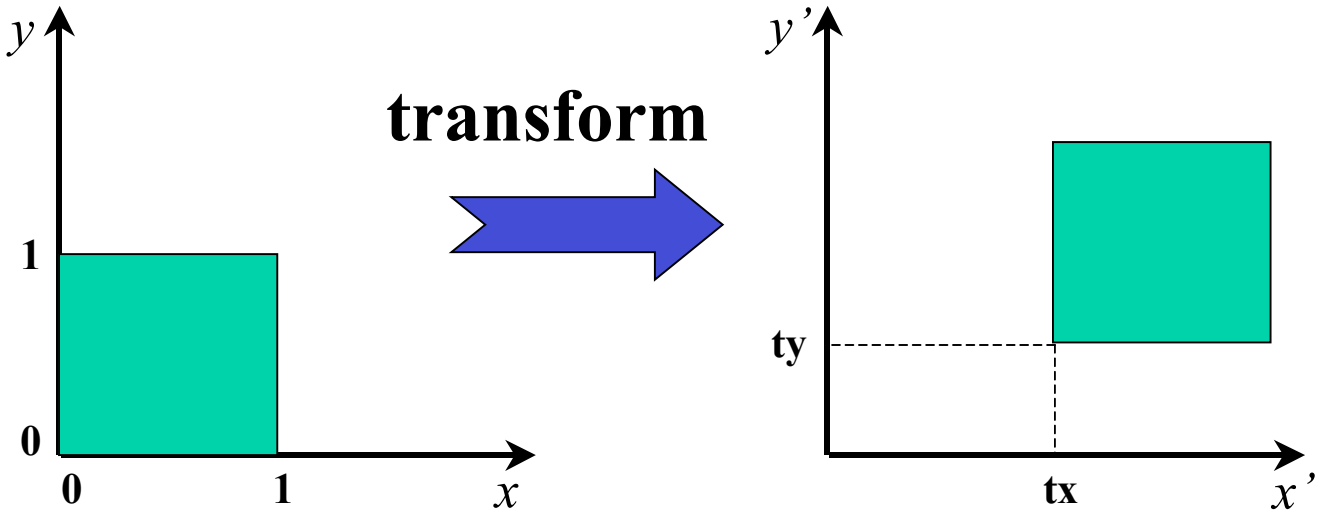
Linear Transformations
(Can be written as matrices)



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{pmatrix} \text{M(params)} \end{pmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Robert Collins
CSE598G

Translation



$$x' = x + t_x$$
$$y' = y + t_y$$

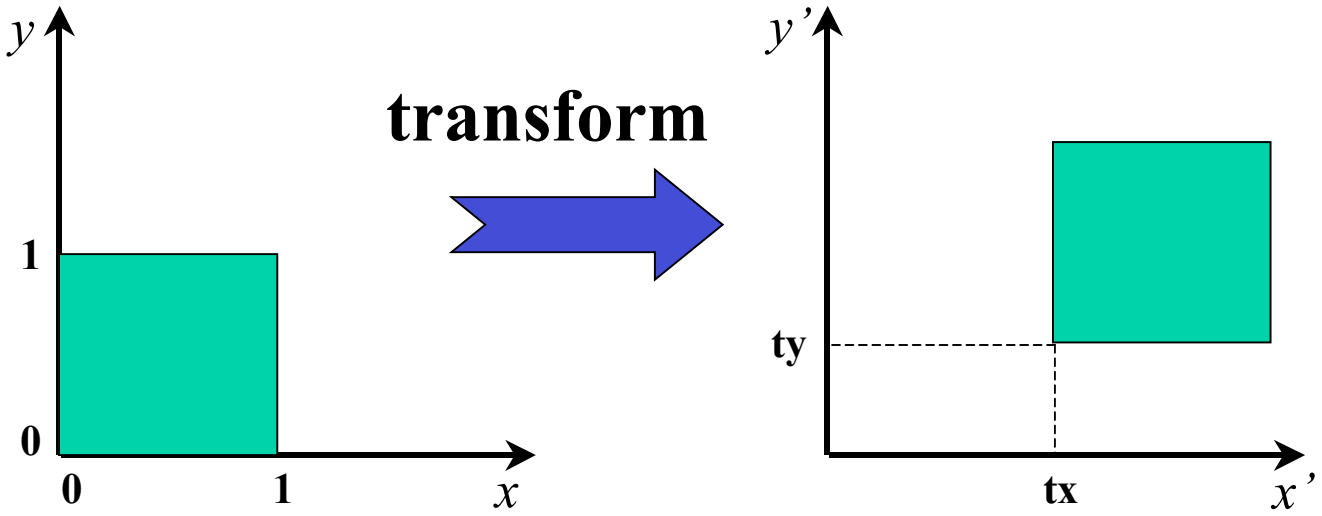
equations

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

matrix form

Robert Collins
CSE598G

Translation



$$x' = x + t_x$$
$$y' = y + t_y$$

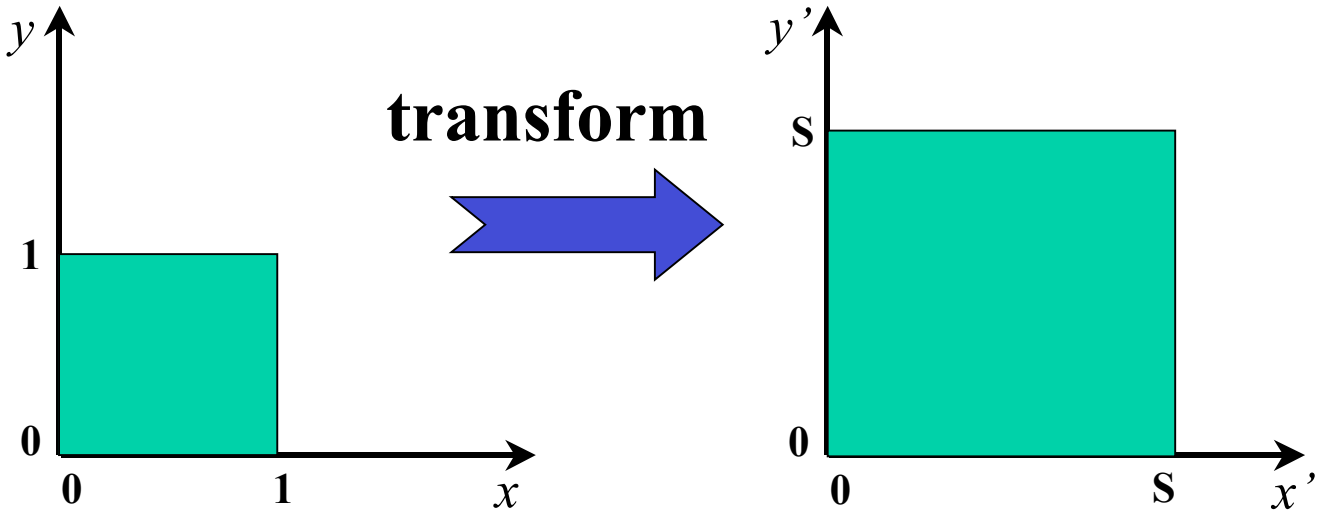
equations

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

matrix form

Robert Collins
CSE598G

Scale



$$x' = s\,x_i$$
$$y' = s\,y_i$$

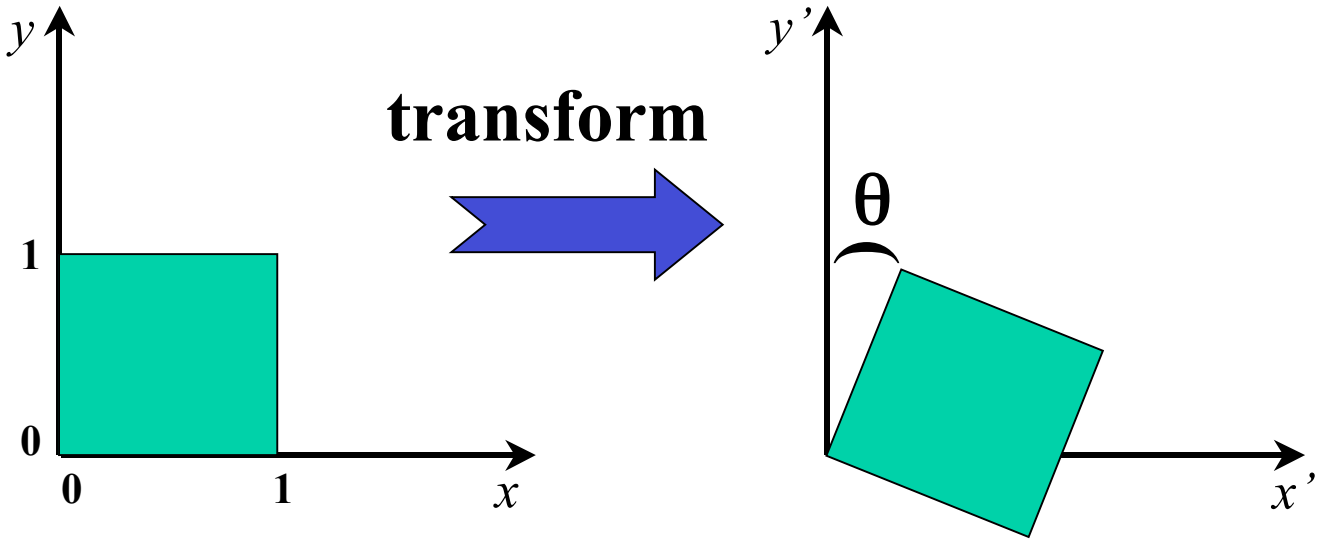
equations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

matrix form

Robert Collins
CSE598G

Rotation



$$x' = x_i \cos\theta - y_i \sin\theta$$
$$y' = x_i \sin\theta + y_i \cos\theta$$

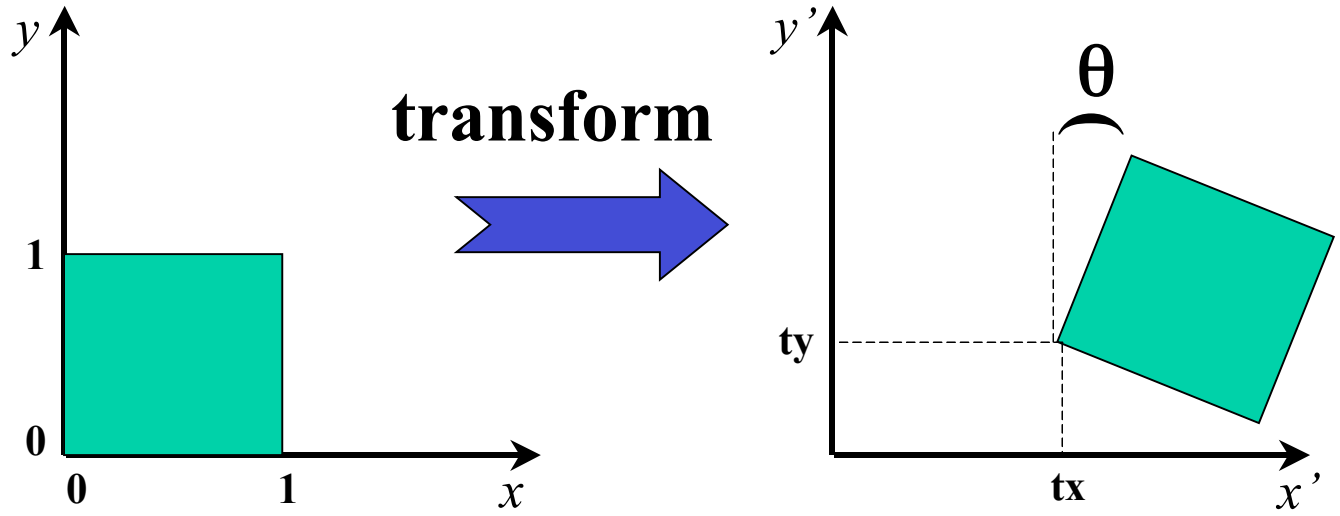
equations

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

matrix form

Robert Collins
CSE598G

Euclidean (Rigid)



$$x' = x_i \cos\theta - y_i \sin\theta + t_x$$
$$y' = x_i \sin\theta + y_i \cos\theta + t_y$$

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

Robert Collins
CSE598G

Partitioned Matrices

$$\begin{bmatrix} x' \\ y' \\ \hline 1 \end{bmatrix} = \left[\begin{array}{cc|c} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ \hline 0 & 0 & 1 \end{array}\right] \begin{bmatrix} x \\ y \\ \hline 1 \end{bmatrix}$$

$$\begin{bmatrix} p' \\ 1 \end{bmatrix}_{3\times 1} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}_{3\times 3} \begin{bmatrix} p \\ 1 \end{bmatrix}_{3\times 1}$$

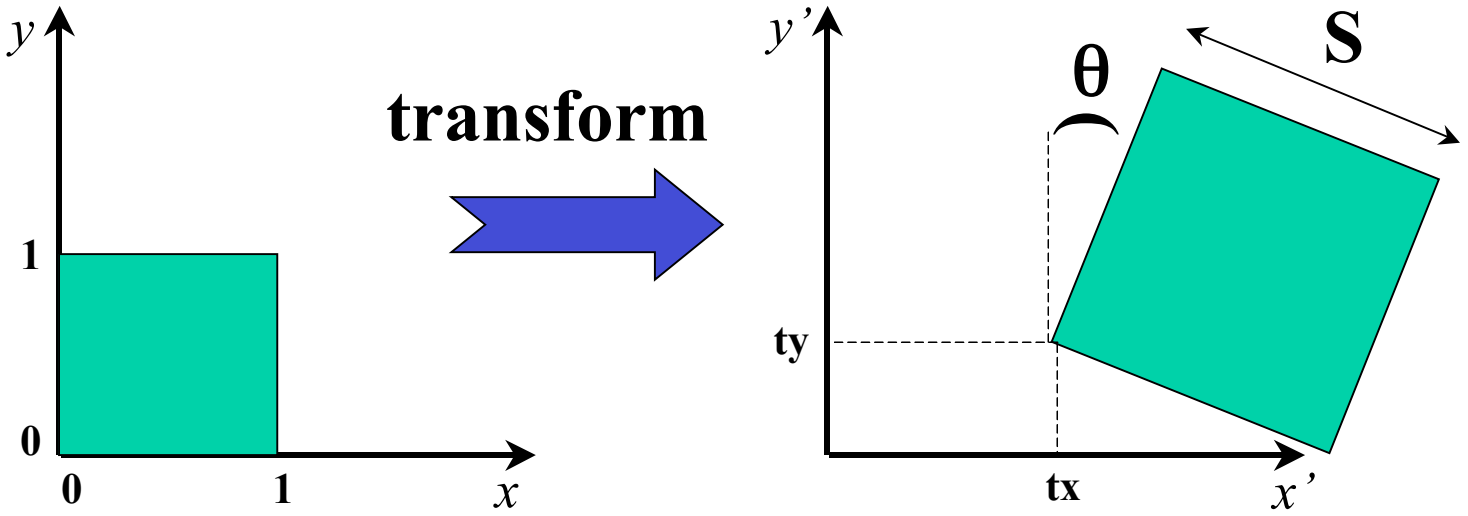
matrix form

$$p' = Rp + t$$

equation form

Robert Collins
CSE598G

Similarity (scaled Euclidean)



$$p' = sRp + t$$

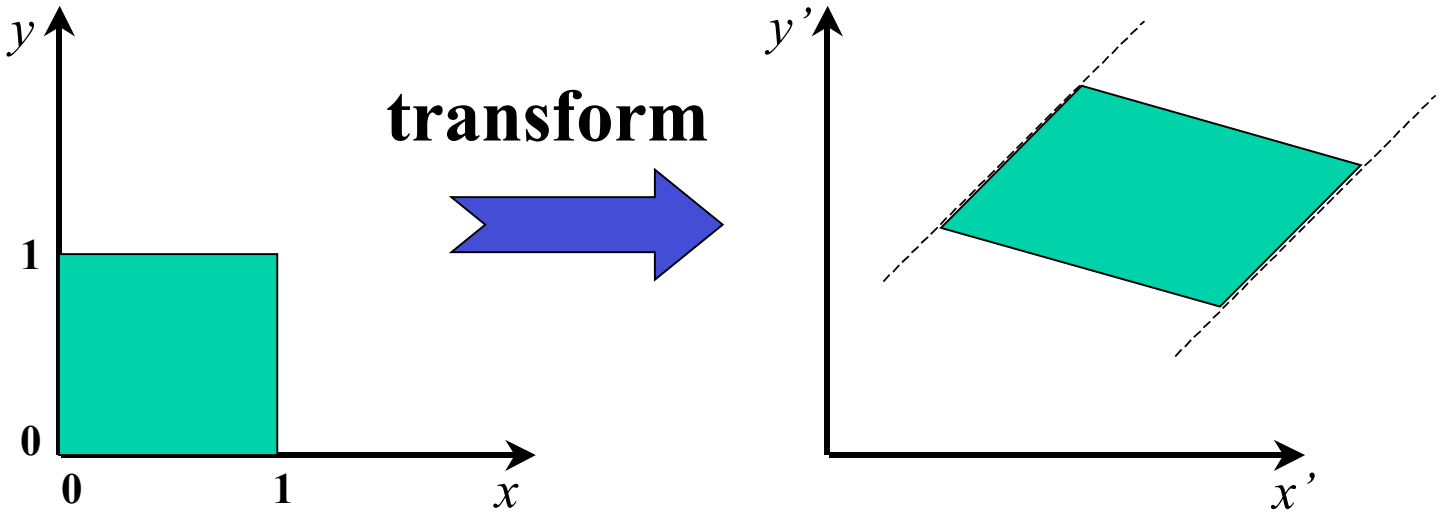
equations

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

matrix form

Robert Collins
CSE598G

Affine



$$p' = Ap + b$$

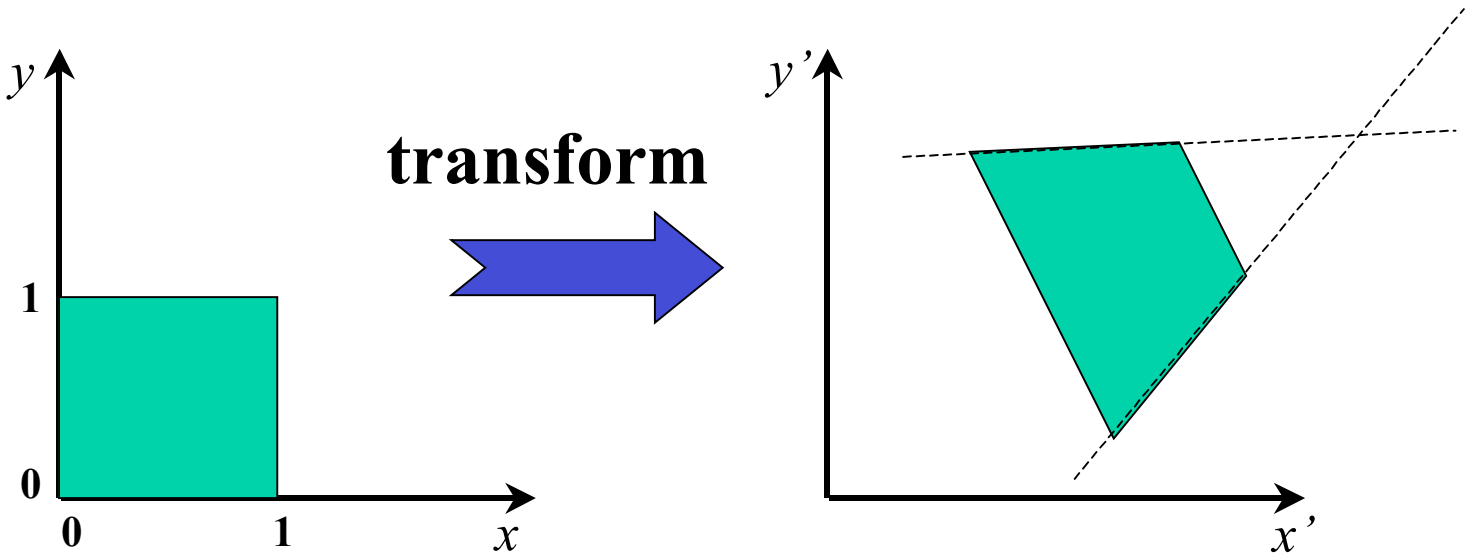
equations

$$\begin{bmatrix} p' \\ 1 \end{bmatrix} = \begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

matrix form

Robert Collins
CSE598G

Projective



Note!

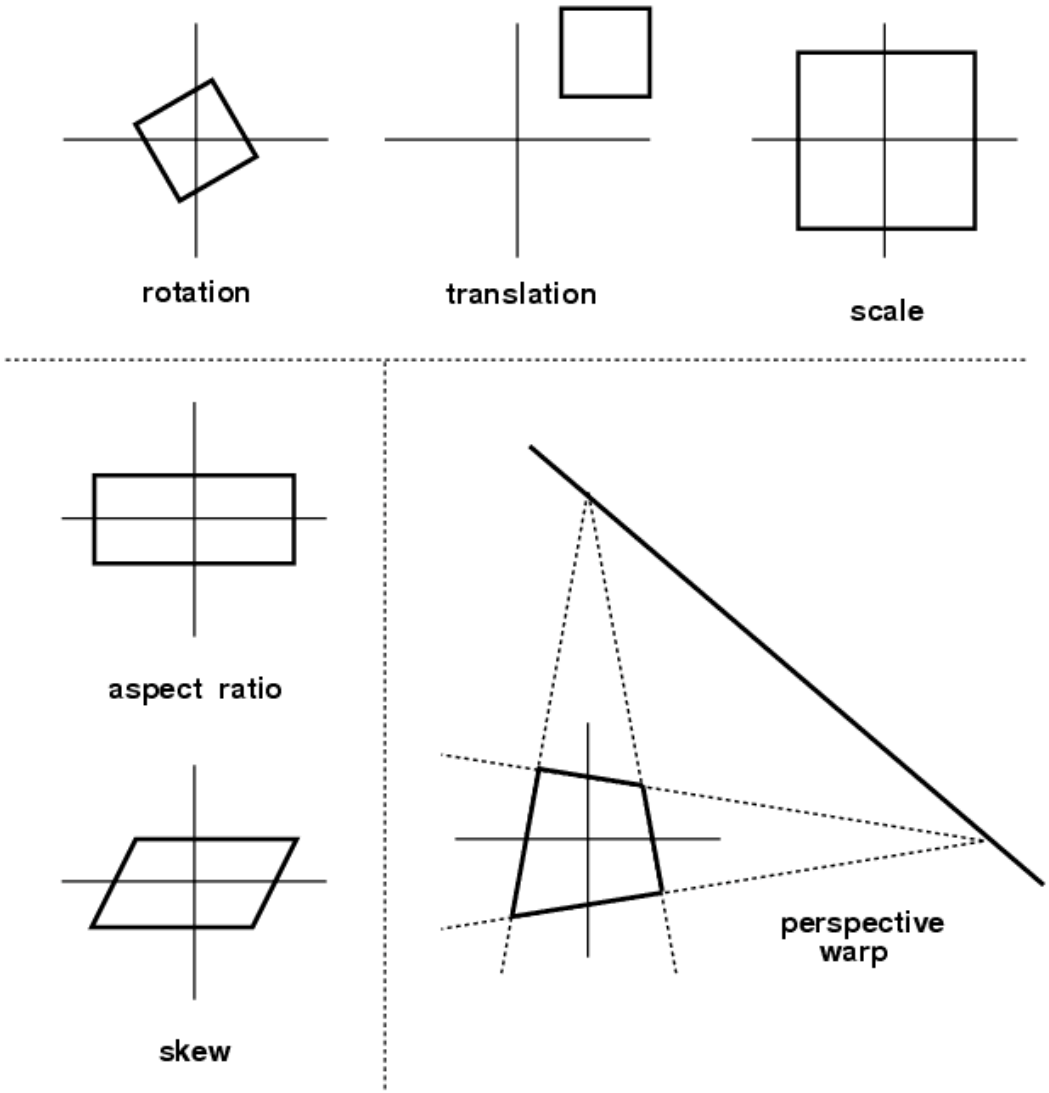
$$p' = \frac{Ap + b}{c^T p + 1}$$

$$\begin{bmatrix} p' \\ 1 \end{bmatrix} \sim \begin{bmatrix} A & b \\ c^T & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

equations

matrix form

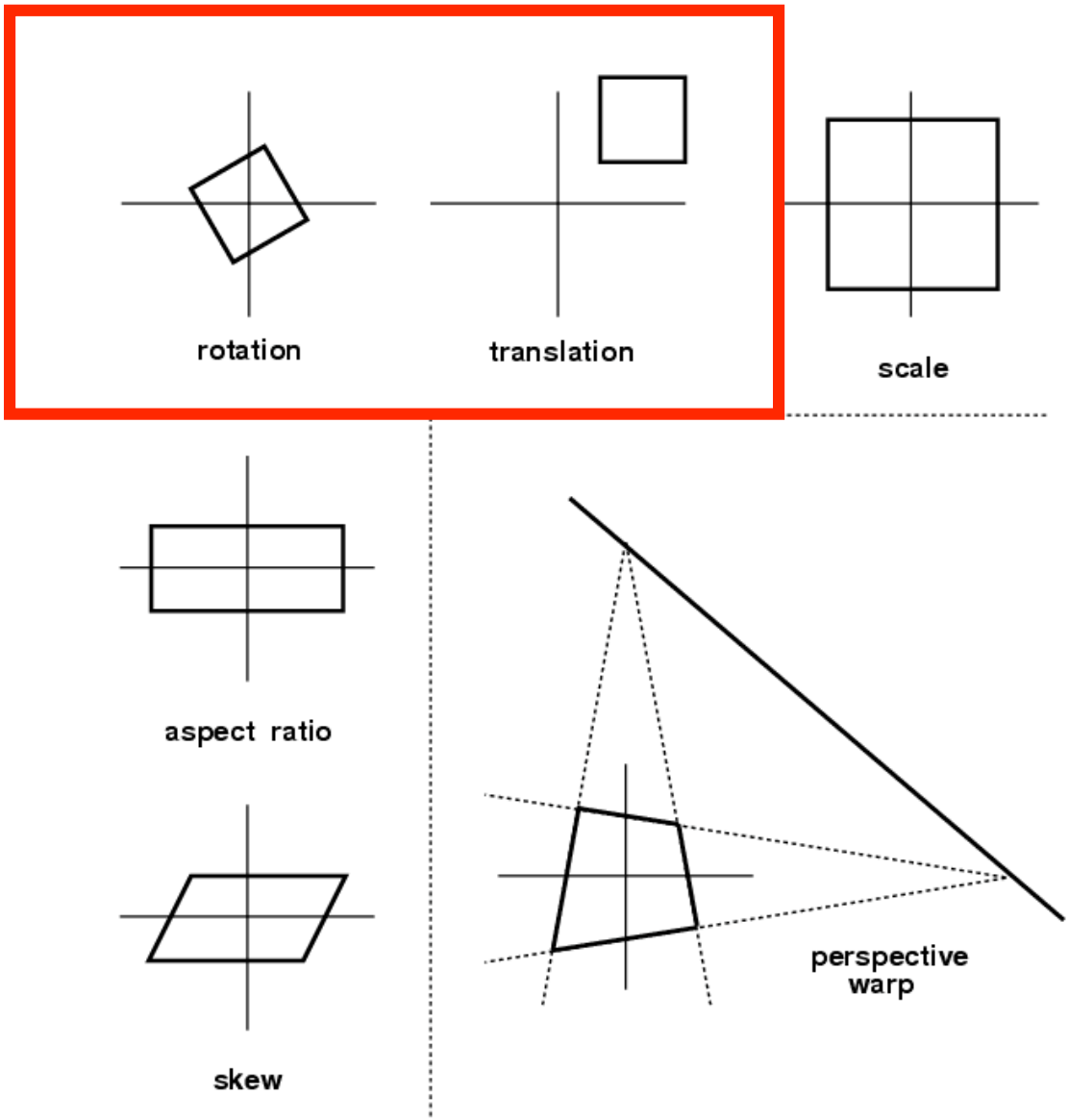
Summary of 2D Transformations



Robert Collins
CSE598G

Summary of 2D Transformations

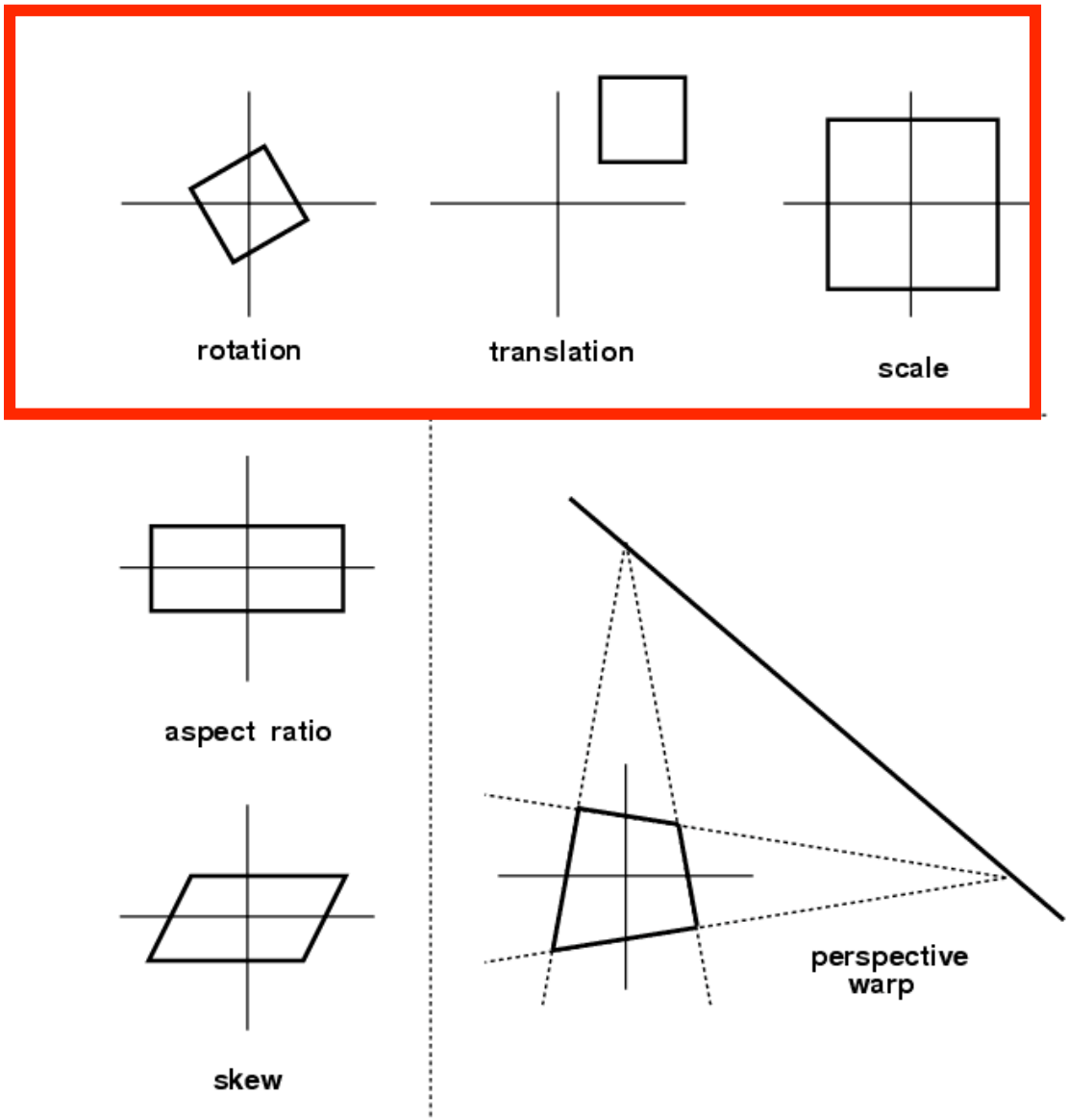
Euclidean



Robert Collins
CSE598G

Summary of 2D Transformations

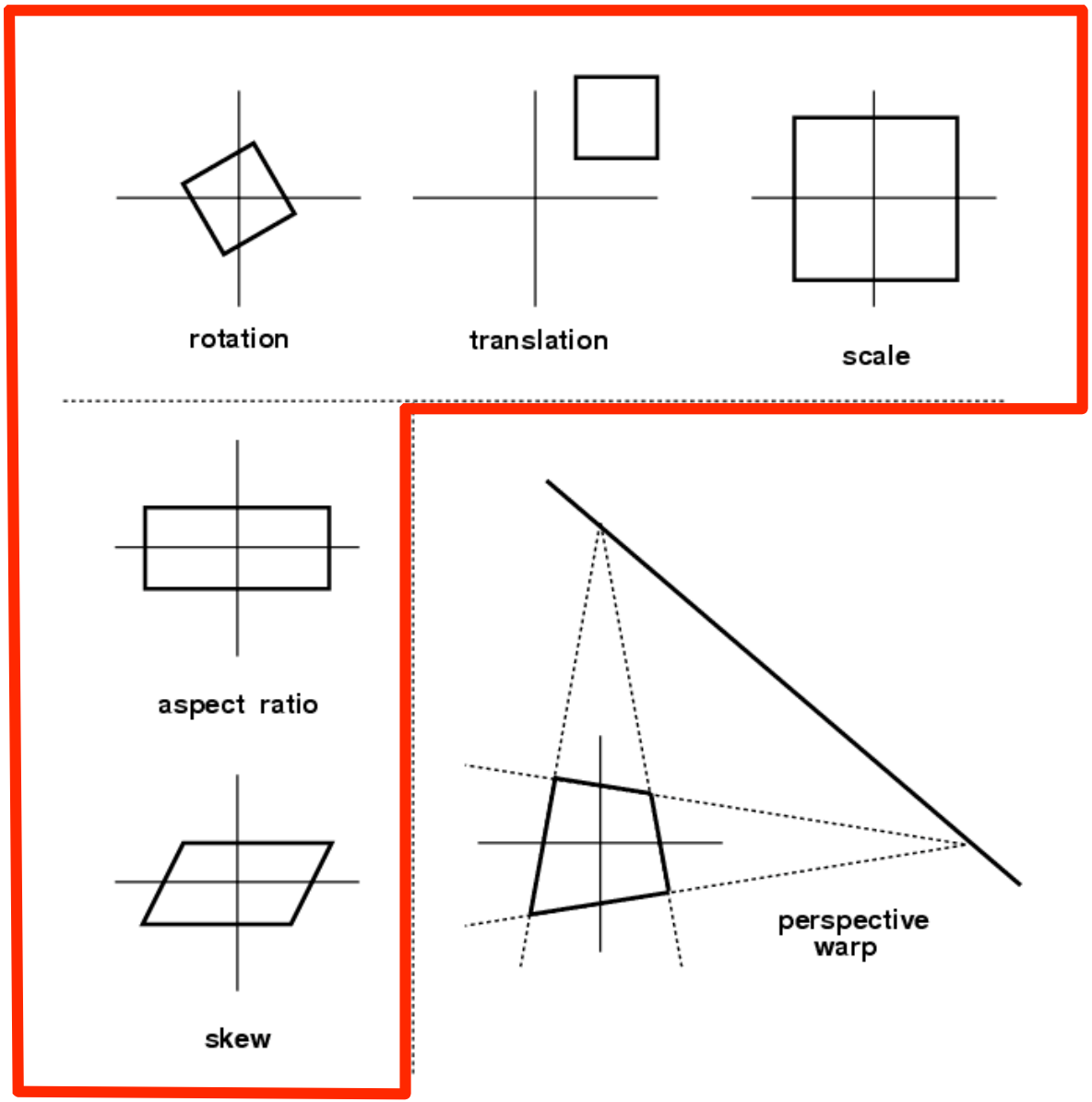
Similarity



Robert Collins
CSE598G

Summary of 2D Transformations

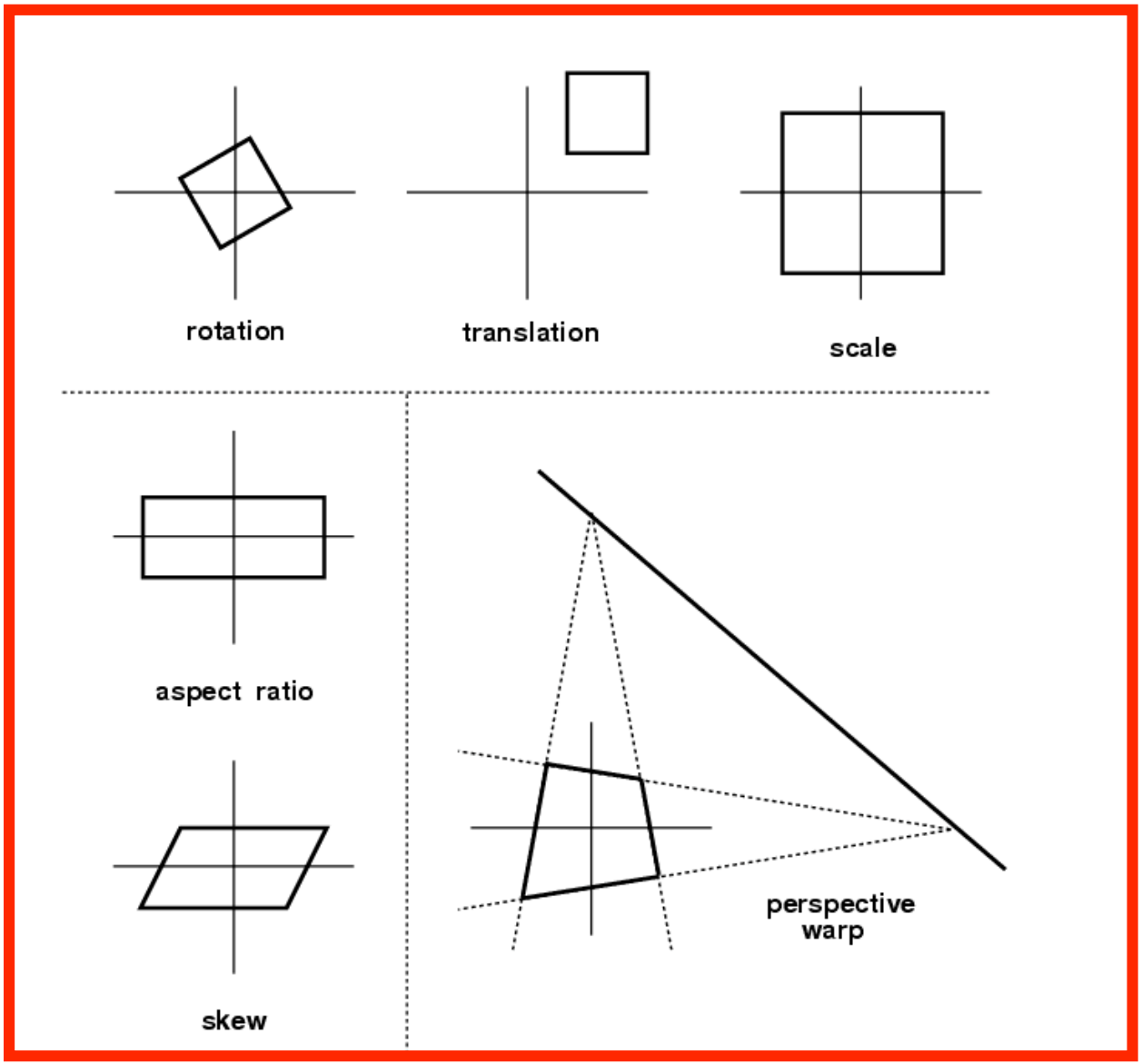
Affine



Robert Collins
CSE598G

Summary of 2D Transformations

Projective



Robert Collins
CSE598G

Summary of 2D Transformations

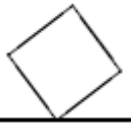
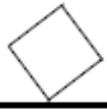
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \boldsymbol{H} \end{bmatrix}_{3\times 3}$	8	straight lines	

table from R.Szeliski

to be continued...