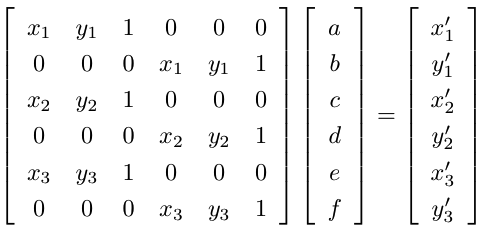
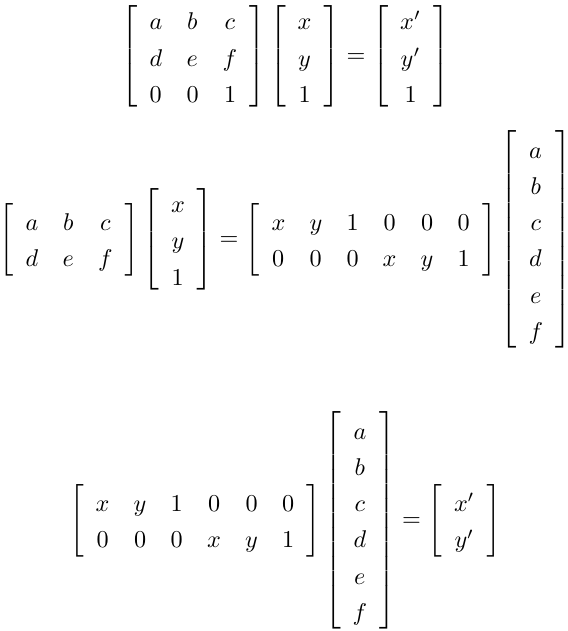
**Estimating a Linear Transform**



• 6 coefficients for an affine transforma3on

-> 6 rows of matrix

-> 3 correspondences needed (x and y give one row each)

**Surface orientation**: color, texture, and reflectance properties (Lambertian vs specular)

**Flood Fill**: Depth-first Traversal (self)

– If self marked, stop!

– Else!

• Mark self !

• For each neighbor!

* Depth-first traversal (neighbor)

**Computer Vision**

– How are images represented?

– How are images formed?

– What kinds of tasks are we interested in doing?

- How do you find what you are looking for?

Morphological operators, Binary image analysis, gradients, general filter responses (mask, kernel, integral image), corners and key points

Different points of view about tasks in computer vision: algorithmic, signal processing, and machine learning

**SIFT** - detect and describe local features in images: corners and dots

Scale Space Octaves - Every time the width of your Gaussian doubles, down sample the image

Gradient Orientation Histogram

• Make a histogram over gradient orientation

• Weighted by gradient magnitude

• Weighted by distance to key point

• Contribution to bins with linear interpolation

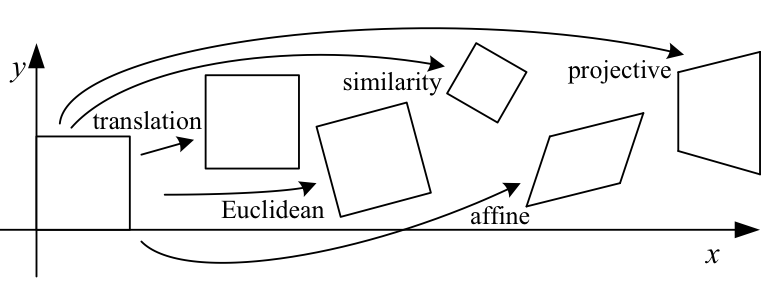
Computing SIFT Descriptor

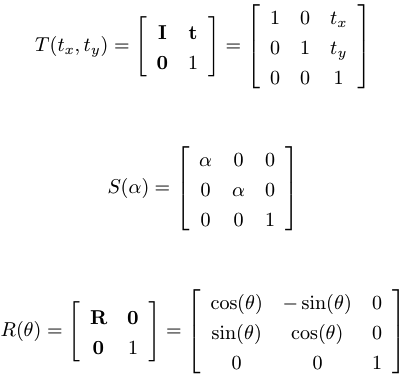
• Divide 16 x 16 region surrounding keypoint into 4 x 4 windows

• For each window, compute a histogram with 8 bins

• 128 total elements

• Interpolation to improve stability (over orientation and over distance to boundary of window)





**Transformation: Projective**

• Also called: Perspective.

Most general linear transformation

• What changes:

– The overall location

– Orientation wrt coordinate axes

– Distances

– Angles between vectors

– Parallel lines no longer parallel

• What stays the same:

– Straight lines stay straight

**Classification with Adaboost**

• Training:

– Given a pool of “weak learners” and some data,

– Create a “boosted classifier” by

– choosing a good combination of K weak learners and associated weights

• In our case “Train a Weak Learner” == Choose a feature to use and which threshold to apply

**Using Features for Classification**

How do you know which feature to use?

-Try them all and pick the one that gives the best result

-Then, choose the next one that does the next best job, emphasizing misclassified images

• Each threshold on a single feature gives mediocre results, but if you combine them in a clever way, you can get good results

-That's the extremely short version of "boosting"

**Classification Cascade**

• Solution: Use a “cascade” of increasingly complex classifiers

• Create less complex classifiers with fewer weak learners that achieve high detection rates (maybe with extra false positives)

• Evaluate more complex, more picky, classifiers only after the image passes the early classifiers

• Train later classifiers in the cascade using only images that pass earlier classifiers

**Properties of Image Regions**

• “Location”

– Set of Pixels

– Boundary

– Centroid

(center)

– Bounding Box

– Oriented Bounding Box

– Convex Hull

• “Shape”

– Area (Total pixels)

– Perimeter (Total pixels in boundary)

– Orientation (Axis of least inertia)

– Elongation (e.g. ratio of dimensions of oriented bounding box)

– Compactness (e.g. ratio of region area to bounding box size)

– Circularity (e.g. ratio between perimeter^2 and area

• “Appearance”

– Mean brightness

– Brightness variance

– Brightness histogram

– Mean color

– Color histogram

– Texture (some statistics about gradients)

**Brightness Statistics**: Count, mean, variance

Otsu's Method of automatic thresholding: Choose a threshold so that the sum of variance of image values in foreground and background regions is minimized

**Segmentation**

**Region Growing (Merging)**

Initialization = all connected components with shared pixel

value

• Grow Criteria = merge regions/pixels when grey-level value

intensity between adjacent regions is at most 1.

**Region Growing (Watershed)**

Initialization = all minimum values

• Grow Criteria = increment each seed region pixel value and add to it

any pixels that are less than or equal to its pixel value; create dam

when two regions merge

• Stop = when all pixels are assigned to a region

Note: dam points occur when dilation causes regions to merge

• Jaccard Index (aka – Precision, Overlap Ratio)

• A = all pixels in object A

• B = all pixels in object B

• Example:

|A inter B|/ |A ∪ B|

**Lucas-Kanade** (Sparse Optical Flow)

**Types of Tracking:** Tracking by detection, Optical Flow/Dense scene motion, Contour Tracking, Multi-target tracking

**Template Tracking**

• Given: small image patch of something we’re looking for

• Goal: Find the best-match location in the new image

• How: Search in a small window around its previous location

• How to compute a matching score?

• Normalized Correlation Coefficient

• Given: template, initial location x0

• For each image, t=1:N

– Search in a small window around x\_{t-1}

– x\_t is location with highest NCC score

• Challenges:

– Computational Cost

– Getting lost / Drift

– Non-translational motion (e.g. rotation)

– Non-rigid motion (articulation of hand)

– tiD motion

– Changing appearance of real object

• What are the benefits/downsides of using larger templates/search windows?

• Why is rotation and scaling problematic for a template tracker?

• If we update the template as we track, what problems do we solve or create?

• How could we benefit from using a constellation of smaller templates instead of one big one?

**Combining Images**

The Pipeline

Image Sequence

Localize Features - Image Keypoints

Extract Features

Match Features - Features

Estimate Transformation - Correspondence Pairs

Transformations

Choose Frame of Reference

Transform Image - Transformed Image

Transfor Weight Map - Error Image

Find Boundary - Blending Boundary

Combine Images

Mosaic

Three Type: “Copy and Paste”, Distance Transform – Use the pixels from the source image whose center is closest to the output pixel, and Multi-band / Pyramid blending (convolve image with successively larger Gaussians DoG, Minimum Error Boundary

**RANSAC** - RANdom Sample And Consensus

• Robust estimation of parameters in the presence of outliers mistakes & noise)

• Given: Corresponding points, error threshold, number of iterations

• In a loop – Choose a sample of the points

– Estimate the parameters

– Compute the error

– Count number of inliers (error < threshold)

– Keep track of the sample and estimated parameters with most inliers and/or lowest error

• How many trials (S)?

– How many points to estimate parameters? (k)

– What is inlier probability? (p)

– What probability that you find a set of inliers? (P)

– Probability that all points in a trial are inliers? p^k

– Probability that after S trials you will not have found a sample with all inliers? (1 – p^k) ^ S

– Probability of finding a set of inliers that you want? P

– (1 – P) = (1 – p^k)^S

– S = log(1-P) / log(1 – p^k)

\_ Sparse Optical Flow / Feature tracking (Lucas-Kanade)

\_ Dense Optical Flow (Horn-Schunck)

\_ Background modeling, mixture of Gaussians

\_ Kalman Filter

\_ Multi-target tracking

\_ Camera Geometry

\_ Epipolar Geometry

\_ Fourier Transform

1