Initialize

```
Off[General::spell];
```

Outline

Last time

Object recognition overview

Today

Object recognition: finishing up compensating for viewpoint changes
Recognition, background variation, segmentation & learning objects

Variation over view: review

From the previous lecture...

Background context, clutter, and occlusion

- Background/context useful for "indexing"

Background can provide prior information, that could be called "index" cues, to narrow down the space of possible objects to be recognized. E.g see: Oliva et al. (2003), Torralba et al. (2006) (pdf).
One of the first demonstrations of the role of background information for human perception was:


**Background (clutter) as a confound**

How vision handles variation over background (clutter) is challenging, very important, yet poorly understood. Background clutter poses three types of problems: 1) segmentation is difficult because clutter near a target object’s borders produce misleading boundary fragments, 1) because local information is often incomplete for objects in a scene, there can be false positives for a target object, and 3) other surfaces may cover parts of the target object, i.e. occlusion leads to missing features/parts of a target object.

Need a better understanding of local image cues, as well as how high-level models can be used to disambiguate local information

**Natural image statistics:**

Let’s look at the problem of segmentation. The same image of an object appearing at different locations will produce quite different local responses in spatial filters.

Let’s place the antlers (right) on the background below (left) at two different locations.
Location 1 (left) and location 2 (right) are shown below. Your visual system has no problem segmenting the antlers.
But compare the local information in the following image blow ups, and corresponding edge detector outputs for locations 1 and 2.
Although different types of edge detectors will give different outputs, it is difficult to remove the ambiguities of what edge elements to link at the boundaries.

- **Texture-based grouping**

  This illustrates the need to take into account region/texture information for segmentation (e.g. Martin et al., 2004).


  The problem of background and clutter suggests that the visual system can make use of both intermediate-level (grouping of features) and high-level information (familiarity with object domains, such as “antlers”) to select and integrate features, both contours and texture, that belong together.

  In this lecture, we’ll focus on the recognition component of segmentation.

- **Analysis-by-synthesis**

  Feedforward and feedback: Use high-level information to predict input and to compare with actual input

Information from high-level model (in memory) can be used to "explain away" the cast shadow contours.


**Bootstrapped learning of object models in clutter**


http://gandalf.psych.umn.edu/users/kersten/kersten-lab/camouflage/digitalembryo.html

- Occlusion
The solution?

Efficient grouping based on similarity. But that may not be enough. One can also use occlusion information to "explain away" missing features.

Consistent with the Bayesian idea of "explaining away".

Neural evidence for top-down processing--Analysis by synthesis

See “Top-down” pdf notes

Next

- Perceptual integration, perception as "puzzle solving".
- Learning object categories
- Spatial layout

Appendix

- Writing Packages

The basic format is straightforward:

```plaintext
BeginPackage["Geometry`Homogeneous`"]
XRotationMatrix::"usage" =
"XRotationMatrix[phi] gives the matrix for rotation about
```
x-axis by phi degrees in radians
YRotationMatrix::"usage" =
"YRotationMatrix[phi] gives the matrix for rotation about y-axis by phi degrees in radians"
ZRotationMatrix::"usage" =
"ZRotationMatrix[phi] gives the matrix for rotation about z-axis by phi degrees in radians"
ScaleMatrix::"usage" =
"ScaleMatrix[sx, sy, sz] gives the matrix to scale a vector by sx, sy, and sz in the x, y and z directions, respectively."
TranslateMatrix::"usage" =
"TranslateMatrix[x, y, z] gives the matrix to translate coordinates by x, y, z."
ThreeDTOHomogeneous::"usage" =
"ThreeDTOHomogeneous[sx, sy, sz] converts 3D coordinates to 4D homogeneous coordinates."
HomogeneousToThreeD::"usage" =
"HomogeneousToThreeD[4Dvector] converts 4D homogeneous coordinates to 3D coordinates."
ZProjectMatrix::"usage" =
"ZProjectMatrix[focal] gives the 4x4 projection matrix to map a vector through the origin to an image plane at focal distance from the origin along the z-axis."
ZOrthographic::"usage" =
"ZOrthographic[vector] projects vector on to the x-y plane."
Begin[
"Private"
]
XRotationMatrix[theta_] :=
{{1, 0, 0, 0}, {0, Cos[theta], -Sin[theta], 0},
{0, Sin[theta], Cos[theta], 0}, {0, 0, 0, 1}};
YRotationMatrix[theta_] :=
{{Cos[theta], 0, Sin[theta], 0}, {0, 1, 0, 0},
{-Sin[theta], 0, Cos[theta], 0}, {0, 0, 0, 1}};
ZRotationMatrix[theta_] :=
{{Cos[theta], -Sin[theta], 0, 0}, {Sin[theta], Cos[theta], 0, 0},
{0, 0, 1, 0}, {0, 0, 0, 1}};
ScaleMatrix[sx_, sy_, sz_] :=
{{sx, 0, 0, 0}, {0, sy, 0, 0}, {0, 0, sz, 0}, {0, 0, 0, 1}};
(*TranslateMatrix[x_, y_, z_] :=
{{1, 0, 0, x}, {0, 1, 0, y}, {0, 0, 1, z}, {0, 0, 0, 1}};*)
TranslateMatrix[x_, y_, z_] :=
{{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {x, y, z, 1}};
ThreeDTOHomogeneous[vec_] := Append[vec, 1];
HomogeneousToThreeD[vec_] := Drop[vec, -1];
ZProjectMatrix[focal_] :=
References


