COSC 4P78

Vision

Week 9

Brock University

Introduction

So, is it Machine Vision, or *Computer Vision*?

Yes.

Actually, the terminology varies depending on whom you ask. Some sources will make a distinction; others will not. For those that do, Machine Vision typically concerns itself with automated applications and lower-level operations; Computer Vision is more often used for higher-level operations, particularly when related to AI.

For the sake of this lecture, we'll be treating them as interchangeable.

So, what are we talking about?

- The ability to give a robotic (or computer) platform “sight”
- The ability to reason, based on visual information
- The ability to recognize patterns in the environment
- The ability to form associations between appearances and ideas

More basically: anything we can do programmatically that involves a camera

Cameras

- Cameras in machine vision are *biomimetic* (i.e. they simulate a real biological function — eyes)
- Higher resolutions may be included for documenting an area, but lower resolutions are normally used for the actual visual processing
- Images may even be resized before processing
- Most machine vision systems use a single camera, but stereoscopy is also an option
- Depending on the application, it may not use standard RGB
  - Lots of tasks work just fine with greyscale, which is computationally more efficient
  - Some may isolate a single channel
  - Some may not even utilize visible light!
    - e.g. Work on plants might deal with IR
    - Project idea!

Applications of Computer Vision

Besides all of the obvious image analysis we're used to (e.g. medical imaging)...

- Detection of defects in manufacturing
- Interacting with humans
- Avoiding humans
- Coordinating assembly
- Security
    - Project idea!

Why is it hard?

There are reasons too numerous to cover, but include:

- Memory requirements
- Throughput
- Oftentimes going over the same image multiple times
- Feature extraction
  - Isolating areas of concern to perform feature extraction
- Noise, shadows, and other visual artefacts
  - e.g. 'you' doesn't look like 'you' in different lighting
- Loss of precision (and features?) due to resizing/rotating
- Inconsistencies with target shapes (e.g. not everyone has the same face)

Basically, we're trying to get a very low-powered device (that needs to be explicitly told what to do) to replace thinky meat. That's hard!
Can we think of any more reasons?

Basic Components

(An entirely *non*-exhaustive overview)

- Capture
  - Buffering
- Preprocessing
  - Smoothing
  - Reduction of colour space
  - Resizing
  - Translation/mapping of coordinates — e.g. to combat ‘fisheye’ effect
- Shape detection
  - Edge detection
  - Segmentation
  - Blob detection (and recognition)
- Modelling
- Classifiers
  - Object detection/recognition
  - Sometimes multiple classifiers voting (similar to *ensembles*)

Let's look at a sample problem...

Suppose you wanted to let your robot guide people towards an exit in the case of an emergency.

How would it know where the exit was?

What might complicate this problem?

Image Processing

Typically, before the “vision” part can be applied to an image, we first need to transform it to make it more palatable. Techniques might include:

- Resizing
- Reducing or transforming colour space
- Convolution
  - e.g. Gaussian blur
- Brightness/contrast, gamma correction, or other transformations
- Thresholding

This is the portion that typically most closely resembles traditional image processing.

Object Detection and Recognition

Object detection is the ability to isolate where objects are. Recognition implies that you can somehow classify said objects, and potentially infer additional parameters (e.g. orientation, configuration, size, etc.).

- Of course, this first assumes that we even know where to look
- It also implies the ability to identify *non-objects*
  - To ignore noise or garbage in the image
  - To not believe that part of an object is the entire object

This can be harder than it first seems. Consider the basic task of identifying a ball in a picture. How would you go about this?

Edge Detection

The simplest method of identifying where an object might be is to identify where its *boundaries* could be.

- If you're looking for a ball, and you find a circle...

There are numerous algorithms for edge detection, but they typically all rely on the same basic principle:

- Gradual shifts in shading are more likely to be shadows than boundaries
- In some direction, look for *sudden* (or severe) changes in intensity, and assume that marks a boundary
- In order to identify different directions of edges, apply the same behaviour in both the vertical *and* horizontal axes

Of course, there are numerous times when this can fail (e.g. two overlapping sheets of white paper will appear to be a single sheet; a single sheet with a black line going through it will look like two).

Segmentation

Both similar to, and often used in conjunction with, edge detection, segmentation is simply the act of extending or separating portions of an image, to try to leave 'similar' visual information grouped together.

- It might include growing a detected edge (or blob — next slide) to clump neighbouring pixels together
- It might mean splitting portions apart if they're considered too dissimilar

As a *general* rule of thumb, segmentation is typically an intermediate step between basic image processing and classifiers.

Note, however, that there are some advanced variations, which we'll get to in a bit.
(Refer to the site that shall not be named, but, uh, I'm going to link to it, for a series of useful images:
`http://en.wikipedia.org/wiki/Region-growing` and
`https://en.wikipedia.org/wiki/Image_segmentation#Clustering_methods`)

Blob Detection and Tracking

Consider this the conjugate of edge detection.

- Rather than looking for a ball based on the edges seeming round, look for a blob of the same colour (e.g. red) that might pass for a circle
- The shape might not even matter — the very presence of a colour might be sufficient
- This can be particularly helpful after you've already identified the subject you're looking for
  - e.g. if you've spotted a redheaded face, even if they turn their head, you can still follow the red

Can you think of how the *tracking* can be made at all easier after the initial *detection*?

Model-Based Vision

Identifying the portions of an image that are ‘interesting’ typically isn’t enough. If, for example, you’re looking for a chair, you likely won’t be satisfied by simply finding ‘lines’. Instead, you’ll probably have an idea of what, specifically, you’re looking for

- You *can* have a 3D model of what you’re looking for, and then try to determine if the lines you’re seeing could be matched up to some orientation of it
  - This requires *far* more processing, but potentially less storage
- You’re more likely to have a collection of multiple 2D images of the object, from different views
  - “Images” might be too generous here; it’s quite possible you’re storing the results of some form of feature extraction or manipulation of the images, and comparing against comparable results from the camera’s images

Hypothetical

Suppose you have a security camera. You want it to notice whenever a human enters its view. How could you approach that?

- Try to match human-y shapes?
- Colour or greyscale?
- Would the environment/setting matter at all?

Basically, what I'm getting at is that you have one *very* important fact helping you here. Can we think of what it is?

Hint: `http://code.google.com/p/scoialrobot/wiki/Tracking`

Motion Vision

Suppose, now, that you have a *moving* image — that is, the entire scene is moving; not just a single subject within it.

- The simple property of being different from the last frame is no longer enough to identify a subject
- *If* we know the nature of the motion (e.g. "camera rotated 5 degrees to the right"), we may be able to use that to our advantage
  - If we know how far things in the background should move, then objects that move a different amount might be of interest to us
- If we *don't* know how far the camera moved, we may still be able to identify the background
  - We'll be relying on both the assumption that the background contains adequate features to be able to match up similar components across frames, and the fact that those components will tend to move together

Optical Flow

Optical flow is an interesting technique that can hint at motion, *without* even having to first identify objects within the field.

- It's based on the idea of comparing the changes of intensity between two frames
- It yields a vector field that shows perceived motion in each location

Note that there are several other alternatives (phase correlation, etc.)

`http://www.societyofrobots.com/programming_computer_vision_tutorial_pt4.shtml`

Optical flow has quite a few interesting applications, ranging from helping a drone determine its own altitude/velocity (when the other is known) to optical mice!

- Seriously, think of how hard motion-tracking would be, with sufficient responsiveness for a mouse, without it

Stereo Vision

Stereo vision is precisely what it sounds like: vision using two images of the same scene

- Traditionally, this implies that the two images were from comparable cameras
- Though different cameras may often be employed for the same scene (e.g. one for colour and one for infrared), this is typically considered a separate application
- If the same shape is identified on both images, and the distance between the cameras is known, basic range estimation may be possible
- Alternatively, as with high resolution imaging, the stereo camera might simply be for offline viewing, with only one of the images being used for machine vision

Pose Estimation

In the event that one is recognizing models, it may be possible to then infer the orientation of said models

- If we make additional assumptions about said models — e.g. the leg bone's connected to the knee bone... — then we may be able to match up a skeleton
    - Note that, in this context, 'skeleton' might actually be more vague than it sounds. We're just about a combination of rigid bodies and connection points. Of course, this describes a human body, but it might also represent any other complicated structure that can have different configurations
- This will generally require a 3D model of the skeleton, and will probably not work terribly well if the subject can't easily be matched up to the expected design — anything from amputation to wizards' robes could potentially throw it off

Facial Recognition

Facial recognition could actually apply to any of three related problems:

- Recognizing that *a* face is in sight
- Recognizing the expression on the face
- Recognizing that the face belongs to a specific person

Though this is true of all of the other recognition/detection topics, facial recognition (and other complicated detections) tend to be particularly susceptible to problems due to *occlusions*.

- Can we think of some common occlusions?

Facial Recognition
(cont)

Simply recognizing *that* there's a face isn't terribly difficult. It's the same basic process as any other feature extraction and detection.
Haar cascades can be one particularly handy approach.

However, recognition the configuration of the face (mouth open vs closed, for example), requires not only identifying the face, but also identifying its specific features.

For the best results, this means mapping a 3D model/wireframe of a face onto the subject.

- Mouth has to match up with mouth, cheeks with cheeks, etc.
- Of course, if the face doesn't have any special markings artificially added, and the model isn't unique to *that* subject, then the generality could cause problems

Alternatively, an alignment of a more basic model is also possible (refer to the links at the end)

Let's pull back a bit

Thus far, we've looked at some very ambitious topics. However, from a strictly numbers standpoint, *most* machine vision isn't nearly that ambitious.

- Some robots use cameras with only a single horizontal line
    - This can actually be more than adequate for some blob tracking, or when simply aligning an arm with a component being assembled
- Oftentimes, we'll 'cheat' by complementing our perception with other sensors
    - For example, first finding where a human roughly is via IR or thermopile, and using that knowledge to confine where we need to perform machine vision

Let's pull back a bit more

- There are numerous far simpler means of identifying subjects
    - Histogram matching is very simple, but can be incredibly powerful
    - Similarly, recognition might go straight from edge detection to other classifiers (e.g. neural networks)

Let's pull way back

While we're at it, here's two questions to consider:

- Suppose you *do* want to use a neural network to classify a pattern within an image...
  - You'd do that... how? The whole image? The centre? ...help?
- Did I actually explain *Haar cascades* or *Haar-like features*?
  - Sometimes you won't get a *great* classifier, but that doesn't mean you can't get great classification
  - Neat link: `http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html`
  - Also neat:

    `https://tech.fpt.com.vn/language/en/face-recognition-and-a-real-world-application/`

Additional Resources

- https://github.com/auduno/clmtrackr
- http://vimeo.com/29348533
- http://auduno.github.io/clmtrackr/examples/facesubstitution.html
- http://dl.acm.org/citation.cfm?id=1938021 or http://link.springer.com/article/10.1007%2Fs11263-010-0380-4 (view from within Brock if you wish to download)
- http://learn.adafruit.com/creepy-face-tracking-portrait/overview
- http://learn.adafruit.com/raspberry-pi-face-recognition-treasure-box/overview
- http://code.google.com/p/scoialrobot/wiki/Initialpage

Questions?

Comments?

Looking forward to next week?