CATFINDER® | DESIGN & DOCUMENTATION

Task: Determine with variable certainty if cats exist in a video frame sent over HTTP

Assumptions & Constraints:

Hosts can store an entire video frame and its related data in working memory

A frame for a 4K video (today's industry standard) is around 3-6Mb. The machines processing video frames will need to possess ~300Mb of working memory to hold the JVM and the actively-processed video frames and whatever RAM the host OS requires (assumed the lightest *nix distro available). The resource needs per host will drastically increase as the industry moves further and further towards 8K and more defined video formats.

Random pixel lookups of video frames occur in O(1) time by utilizing a certain data structure

If a frame requires linearly more time to access the n^{th} pixel compared to the 1^{st} pixel, this algorithm slows down to a halt for recognizing cats in the bottom of the frames. In this scenario, a buffer or 16 rows will need to be used to hold the actively processed rows.

Cat faces are aligned with the axes of the video frame

If there is skew in the cat faces within the video frame, something fairly common in kitten videos, this algorithm will need to be restructured.

Cats to be recognized look exactly the same and are of the same size

If there are other reference cat images, the match determination algorithm will need to be parallelized to speed up processing. If there are multiple reference images of various sizes, this will further complicate the solution.

Security, authentication, scalability are not concerns

This solution does not incorporate multithreading or HTTP over SSL. Authentication of users should be considered to ensure the wrong cat video frames are not sent to this service.

The reference image will not change

The reference image is hardcoded into the solution, changing this may result in the solution being taken offline for a brief moment reducing overall uptime metrics, an important metric for Tier-1 services.

Whitespace beneath reference cat image is intentional

If whitespace can be ignored, the processing time of the algorithm can be reduced by ~19%.

A non-cat object may be recognized as a cat if the surrounding noise lines up to meet the match threshold

To prevent false positive rates, surrounding noise might need to be analyzed.

Cat Recognition Algorithm (CRA):

Selected Approach:

Progressively scan groups of rows of pixels at a time and sequentially scan the groups horizontally for a cat image that matches the reference image. In this case, groups consist of 16 rows of pixels that are scanned.

To determine match, a more rudimentary approach was taken: the number of "pixels" matching the reference image will be utilized to determine match rate. For example, if 200 out of 240 pixels match the reference image, that is considered to be an 83% match. This will be checked against the incoming threshold match value to determine if block of pixels is a cat. This allows easy configurability for the end user and is relatively simple to implement, maintain and scale.

Rejected Approaches:

To scan video for cats, features of the cat/input image are searched throughout the video frame. For example, the video frame will be scanned for the nose or eyes of the cat and then the algorithm will determine if surrounding pixels represent a cat.

Reason for rejection:

Shouldn't be significantly faster according to calculations. Entire video frame will need to be scanned to identify any example of a feature and if video is noisy, this can lead to a lot of false positives. Only benefit of this approach is the lower working memory usage. This approach might work better for low noise videos. This approach can work drastically better if larger features that are more likely to have a low false positive rate are utilized for scanning.

To determine match against the reference image, noise around certain features can be considered to be less important than other pixels. For example, if there was one extra pixel next to the “cat's“ left eye, that can be considered normal.

Reason for rejection:

Time taken to implement.

Optimizations to Consider in the Future:

When considering a block of pixels, parallelize the pixel matching process across multiple threads. This will be incredibly useful if the reference image grows in size.

Input for service can be a video

Video frame extraction was not covered in this solution. The solutions vary greatly from creating your own library from scratch to support a unique video format (for example: .avi files) to utilizing an off the shelf solution (for example: https://github.com/artclarke/xuggle-xuggler/tree/master/src/com/xuggle/xuggler/demos)

Real-time cat detection:

If real-time detection is needed, there is a great way to carry out the task by utilizing the cv2.VideoCapture class and building around it as shown here: https://www.pyimagesearch.com/2016/01/04/unifying-picamera-and-cv2-videocapture-into-a-single-class-with-opencv/

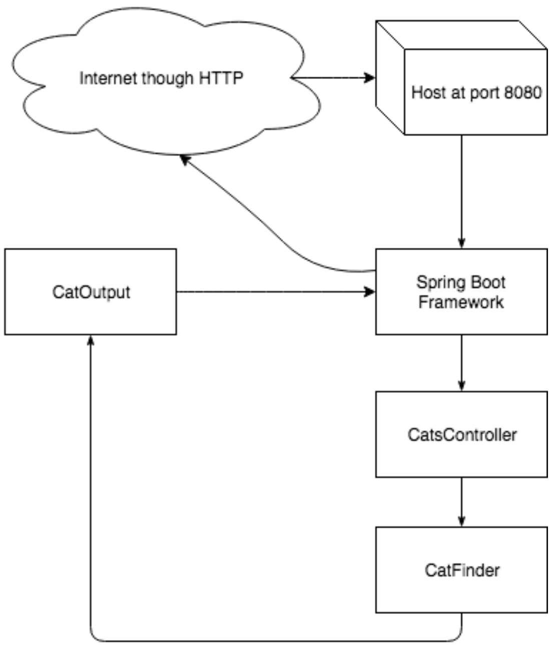
Once frames are extracted, the frames can be sent to multithreaded hosts that can analyze the frames using the algorithm described below. To enable real time detection, the number of hosts required and the performance capabilities needed from each host greatly depends on the resolution of the video as well as the refresh rate of the video. A real-time 2160*p* (16:9 4K) @ 360fps video stream will require significantly more resources than a 240*i* @ 30fps video stream (Note the usage of *i* and *p* suffixes after the resolution).

Implementation

The Spring Boot framework was used to capture the incoming HTTP requests received by the host at the 8080 port (default value which can be configured). Requests made to the "`/findcats`" API endpoint and the "`/findcatswiththresholdmatch`" are routed to the CatsController class by Spring Boot. This controller class calls the CatFinder class to process incoming video frames while passing a threshold match value which should be sent as an `int` by any client calling the host via HTTP.

Incoming video frames are expected to be sent via an array of Strings. This can change with some minor refactoring. If, instead, a giant String needs to be passed, `string.split("\\r?\\n")` can be used to ensure major refactoring is unnecessary). HTTP requests in and out are expected to be in the JSON format.

The project uses Apache Maven for build automation and application start.



Deployment

1) Make a directory to hold the project files
2) Clone the *git* repository using the following command:

```
git clone https://github.com/suryamp/CatFinder.git <folder name created in step 1>
```

3) CD, change working directory to <folder name created in step 1>
4) Run the following command:

```
./mvnw spring-boot:run
```

5) Application can now be reached via `host_ip_address:8080`