2 years ago

Hi Cav. Welcome to the course and good question.

Below, I am scaling down a fan-art image of [Princess Belle](https://proxy.duckduckgo.com/iu/?u=https%3A%2F%2Fi.pinimg.com%2F736x%2Ff0%2F7e%2F96%2Ff07e96c57ba23a69961a4d721b432d46--mona-lisa-smile-princess-belle.jpg&f=1) by a factor of 1.5 with each step:   
      +  scaling down by a factor of 1.5 means "dividing by 1.5" each time  
      +  the width of the images are 400px, 277px, 178px, and 119px  
      +  notice that it isn't until the 4th image that the green circles even begin to line up with some of the facial features

  

  **<<< This one looks like it could contain the frontal-view of a face!**

The mechanics of how cv2's CascadeClassifer.detectMultiScale() function works is really cool, I think.

What's important to realize here is that the haarcascade\_frontalface\_default.xml file defines:  
      +  a selection of a few face-shaped boxes, i.e. tall face, round face, potato head, large forehead, prominent chin, etc.  
      +  relative positions, shapes, and sizes of expected facial-features: chin, forehead, hair, eyes, nose, mouth, and ears

Using the face-classifying information defined by the xml-data, the detectMultiScale() function will:  
      +  examine the target image in multiple passes, while scaling down the image a bit smaller with each new pass  
      +  within each pass, each defined piece of face-data is virtually-superimposed upon the image many, many times:  
          +  every possible location within the image is examined, using each type of face that's defined in the data  
          +  each virtual-superimposition operation attempts to identify any significant correlation between the image and the data

During all of this, the size of the face- and facial-feature data is fixed to be the same size throughout all of classification passes. This data is roughly represented in my images above by the green-lined oval shapes.  
*Note: the green shapes are hand-drawn face+eyes+mouth and, as a result, the overlays are not precisely all the same size there.*

**scaleFactor** -- changing this parameter is what determines the incremental size-change between each pass  
      +  in many cases, smaller values for the scale-factor will increase the success of classifying faces  
      +  however, specifying a too-small scale-factor value will:  
          +  intensify the overall processing and memory requirements of face-detection, yet  
          +  at some point, there's little or no added benefit in identifying "more and better" possible faces

**minNeighbors** -- changing this parameter helps the function decide when to discard a possible face-detection  
      +  when a possible face is:  
          +  partially extending off the edge of the image, or  
          +  substantially viewed from the right- or left-side, or from above or below, then  
          +  at some point, a possible face should be discarded because it's probably not actually a frontal-view of a face  
      +  when we specify a value for minNeighbors, we are tweaking the question:  
         +  *"Is enough of this face visible in the image to successfully classify this as a face?"*