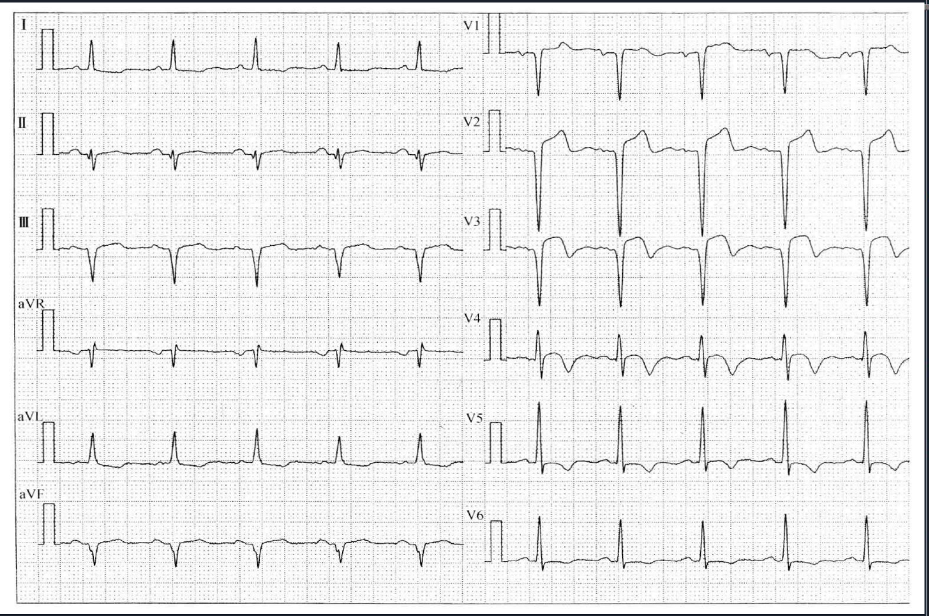
ECG Digitization

Mao, Junyan

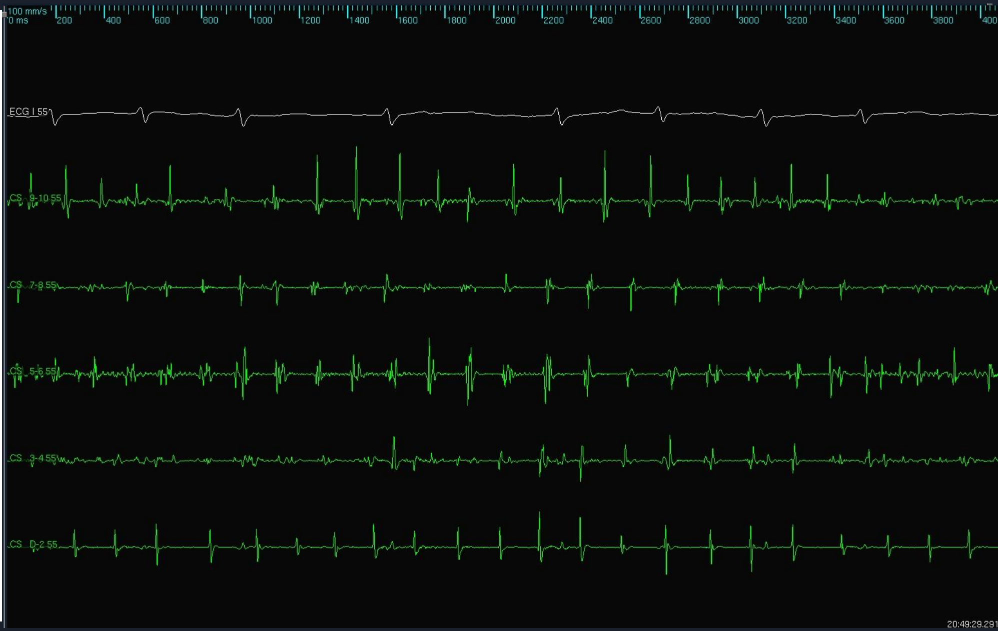
Problem

An Electrocardiogram (ECG) keeps track of the electrical signals in your heart. It’s a common test used to detect heart problems and monitor the heart’s status in many situations. Currently, a lot of ECG records, unfortunately, are only stored in printed form, which cannot be processed or analyzed by a computer algorithm or an AI. While a human doctor would be able to make initial diagnoses on an ECG just based on bare sight, such a ‘noisy’ scanned ECG (example on next slide) would make no sense and is unlearnable to an AI. This project aims at developing a Python-based pre-process procedure, with computer vision techniques, that makes the original ECG and internal cavity ECG more understandable and learnable by an AI.

Example of ECG and internal cavity ECG (respectively)



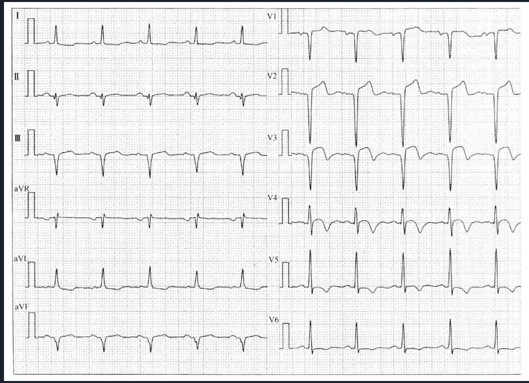
noisy



clean

Problems with scanned ECG

1. The dots in the background are bare noises, which prevent the AI from learning anything, and need to be eliminated.
2. The outside boundary does not help with our analysis, it needs to be cropped out.
3. There are some gaps in a single curve, which will cause the curve to be recognized as 2 parts instead of 1 continuous curve, so they need to be filled.
4. The labels (I, II, V1, V2, …) for the curves also need to be removed before we extract the curves.



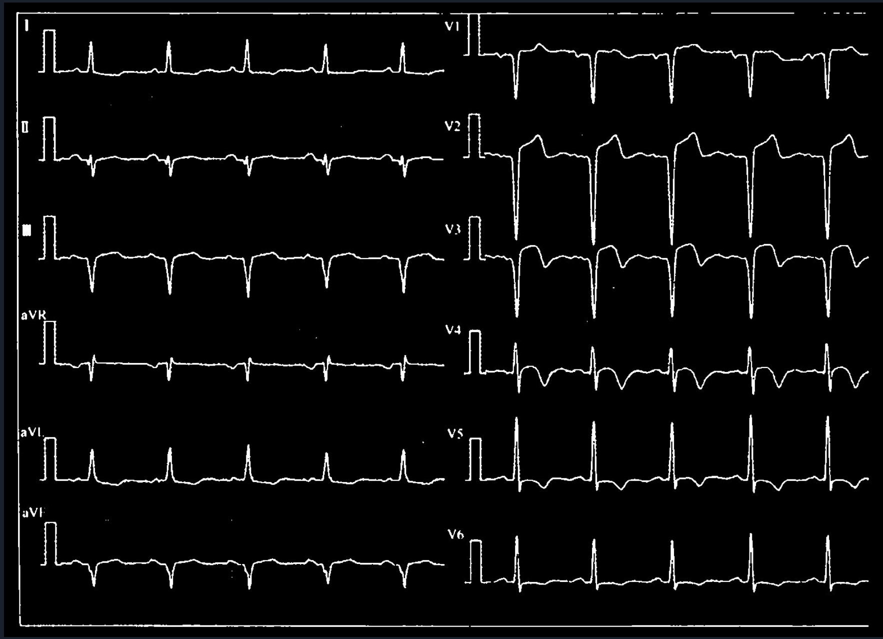
Problem: Dots in the background

- Read the image as grayscale
  - Since color dimension is not needed
- Apply Gaussian Blurring and Median Blurring to the image
  - To reduce image noise and details
- Apply Adaptive Thresholding to the image
  - To further reduce background noise and binarize the image

Result on next slide

Problem: Dots in the background

- We can tell that the background dots are almost all eliminated while preserving the shapes of the curves.
- There are still several white dots in the background, we will deal with that in a later step



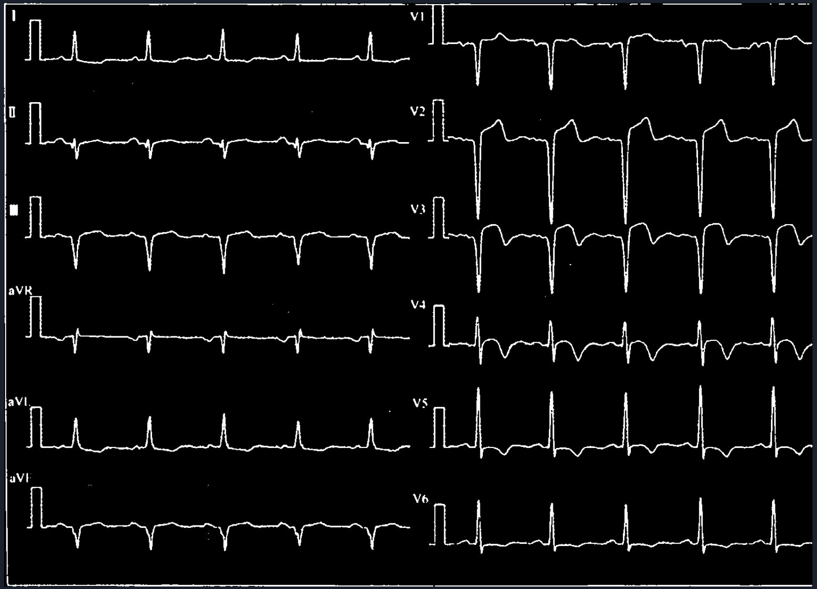
Problem: Outside boundary

- Apply an masking algorithm that approximates the outer boundary, cropping out the empty areas.
- Further crop a few pixels to totally eliminate the boundary.

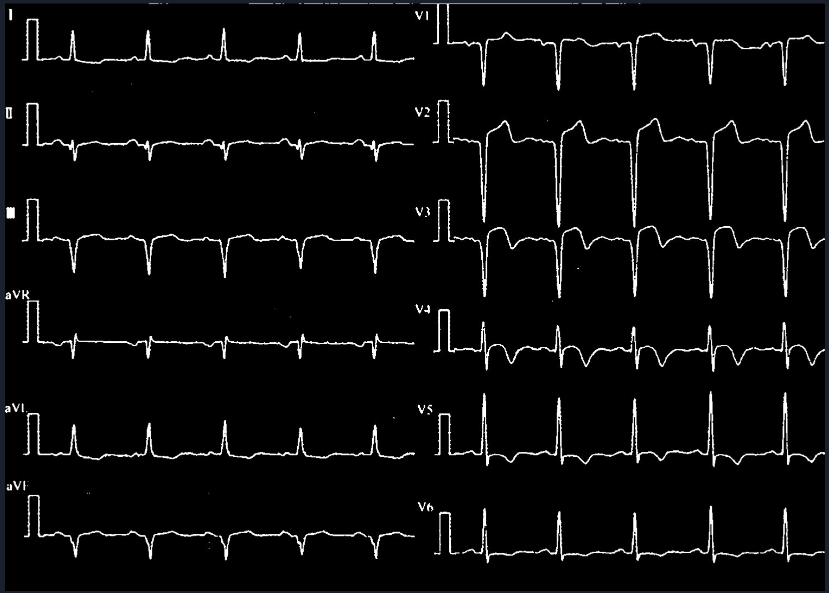
Result on next slide

Problem: Outside boundary

Approximating boundary



Further cropping

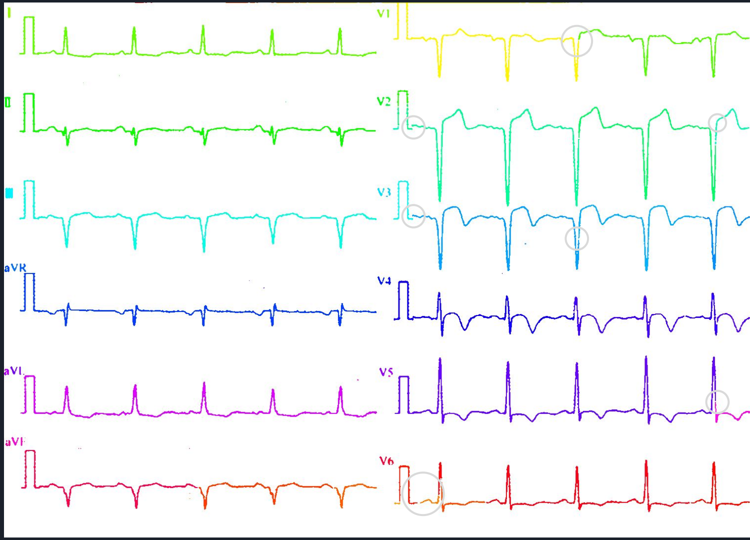


New problem: the highest part of V1 is actually connected to the boundary.

Problem: Gaps in a single curve

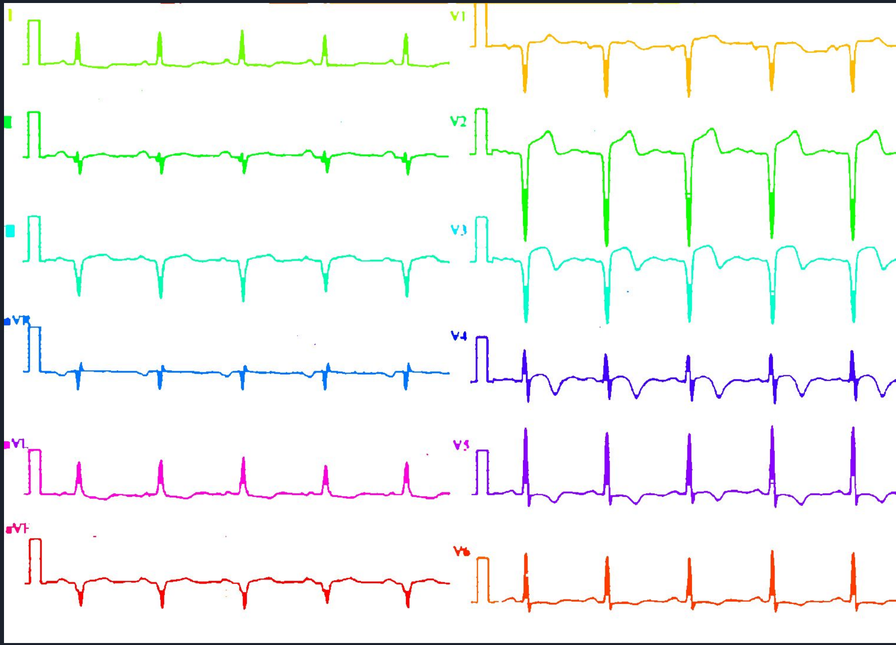
- Apply a labelling algorithm on current binary image.
- We can tell that there are several curves that are identified as two or more separate curves, which is caused by the gap.

Several examples on the right



Problem: Gaps in a single curve

- Fill the gaps by applying dilation + erosion
- Dilation
- Erosion
- Label again
- We can now see that the curves are separated correctly

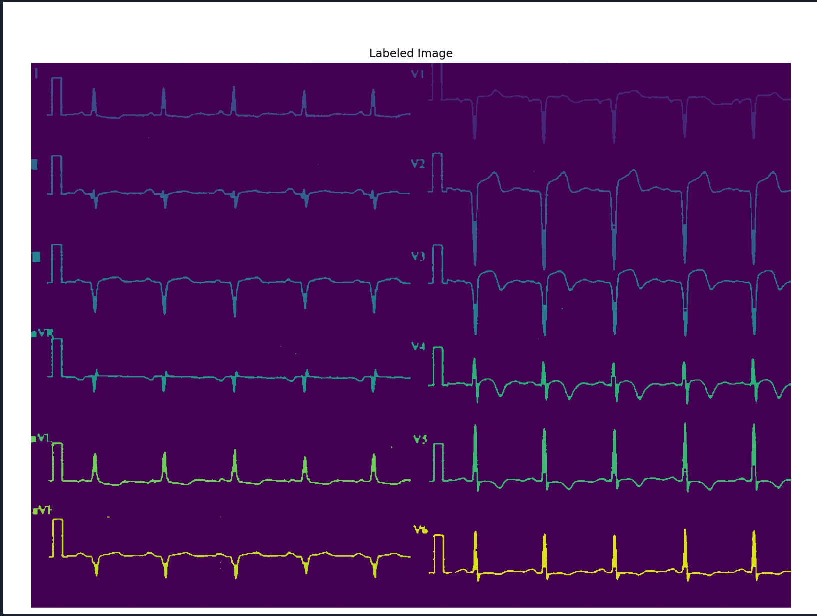


New problems:

1. When doing dilation and erosion, the 'aVR' label was merged into that curve.
2. Some of the peaks are filled due to this process.

Problem: Extracting individual curves

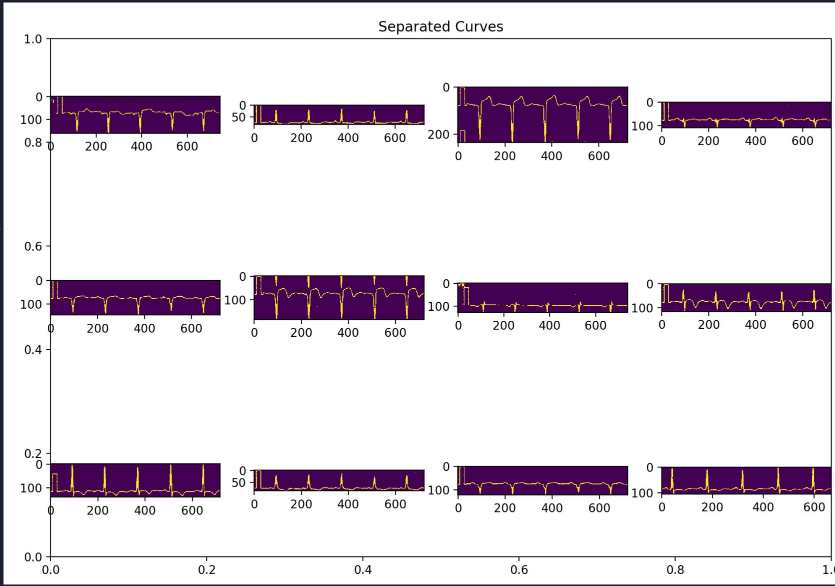
- Apply another labelling algorithm in SciPy's ndimage package to get access to each connected component



Problem: Extracting individual curves

- Apply another labelling algorithm in SciPy’s ndimage package to get access to each connected component
- Filter out small pieces and noises by inspecting the shape of the components. We can now see that 12 individual curves have been extracted.

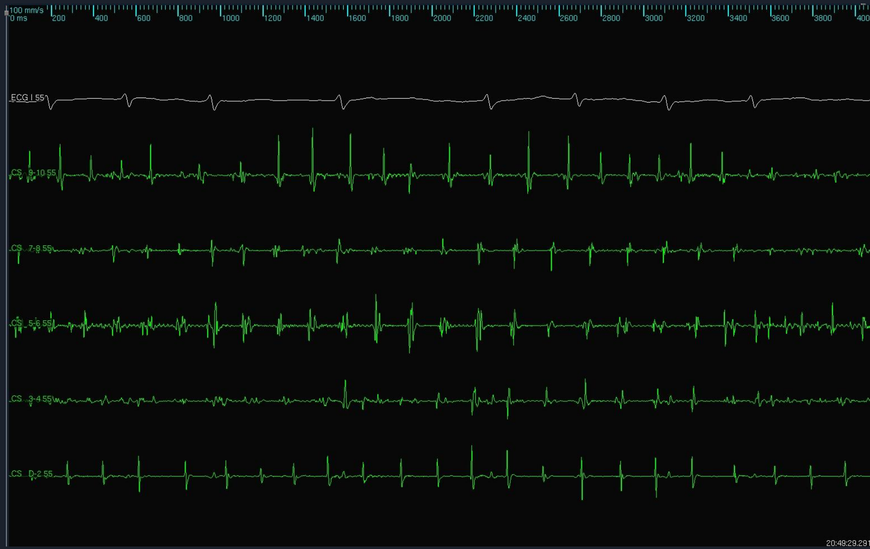
Note: You might see in some examples, there are parts of another curve being shown. This does not mean that they are of the same component. It is because when we are cropping the curve from the original image, we crop the whole area that a curve occupies, which might include some parts of another curve



Now that the individual curves have been extracted. We can continue to do signal extraction and grayscale conversion. This will be introduced later on the example of internal cavity ECG in this presentation.

Internal cavity ECG

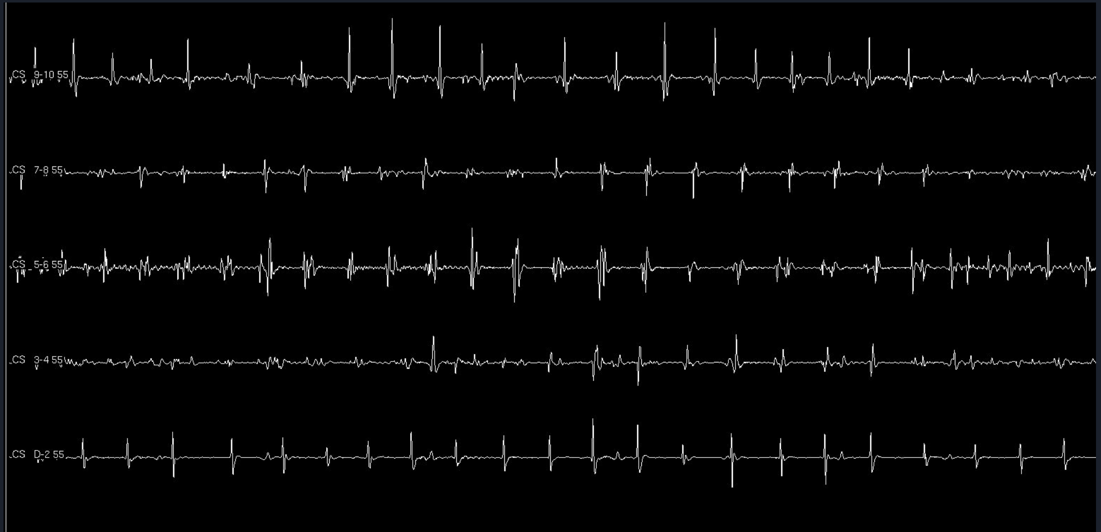
- This is an example of an internal cavity ECG, which are screenshotted from a computer.
- Since the background is solid black, there is basically no additional noise, unlike the scanned ECG.



Step 1: Extracting the curves

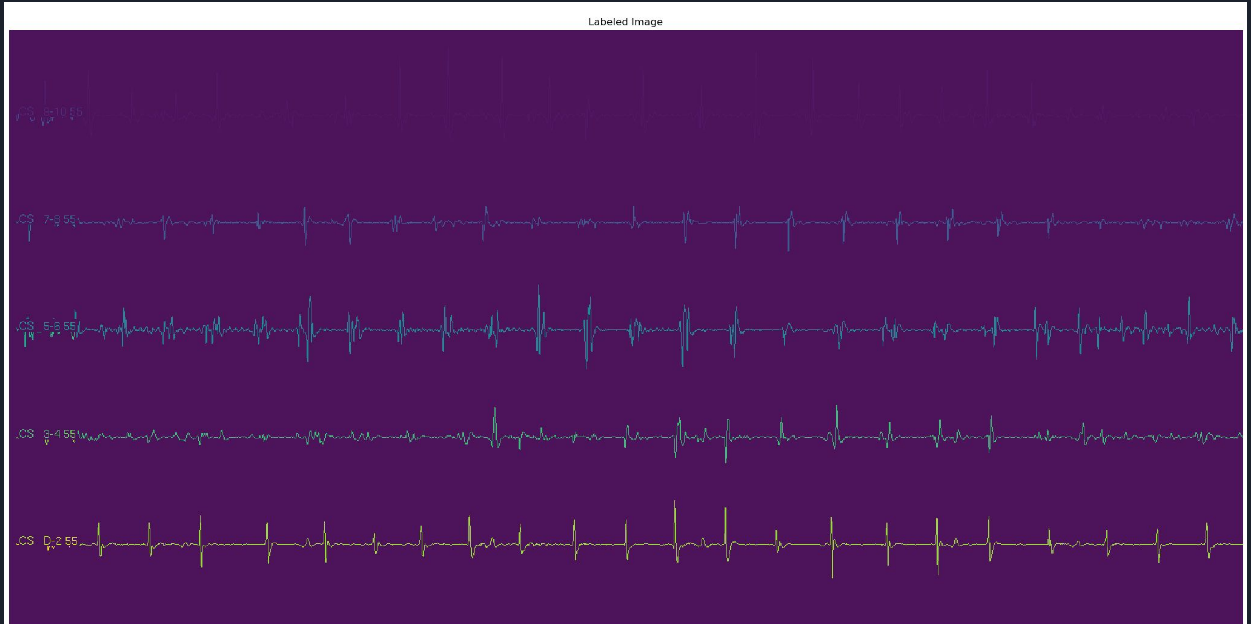
- Read the image as grayscale.
- Since we only want to extract the five green curves, we crop out the scale and the white curve above, and the tag below.
- And we binarize the image to reduce dimension.

Result



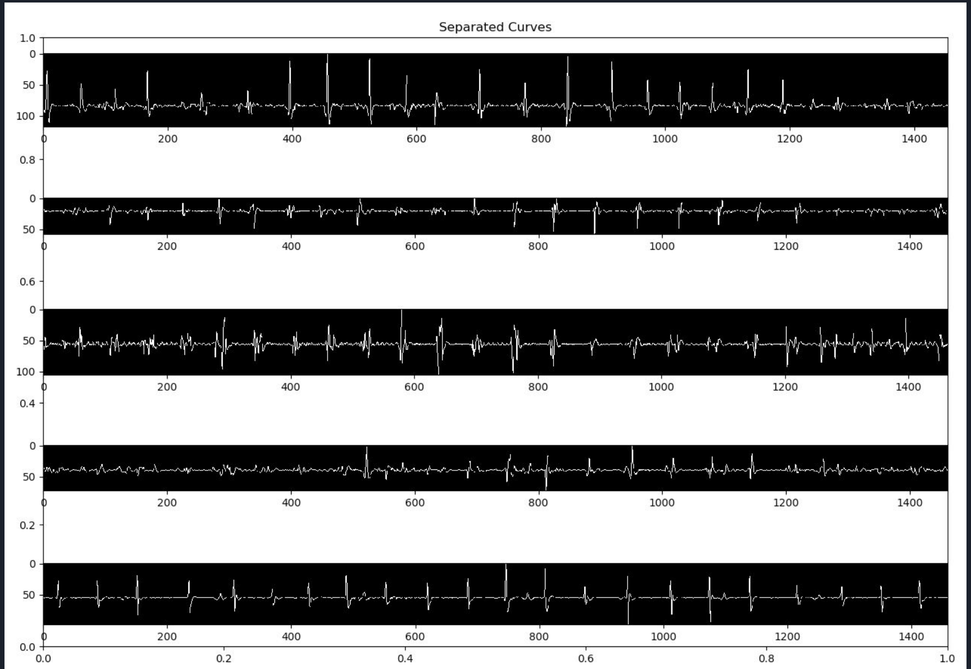
Step 1: Extracting the curves

- Label the image.



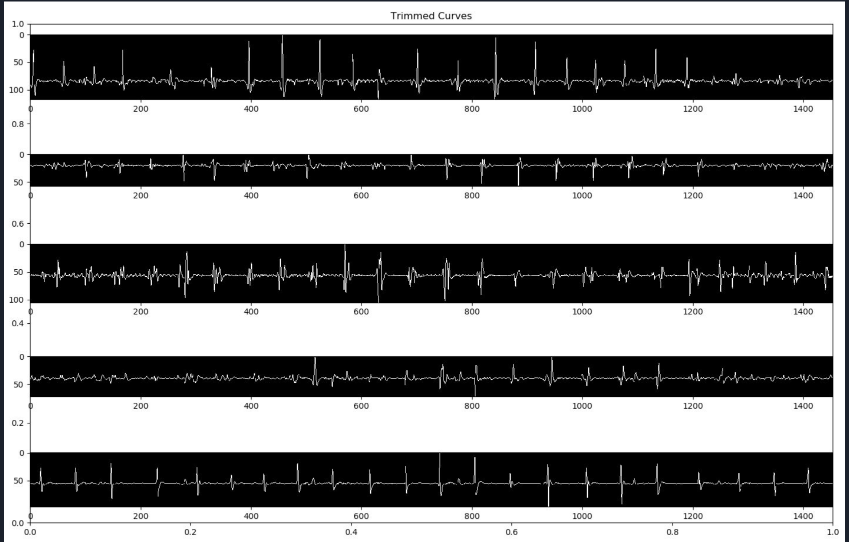
Step 1: Extracting the curves

- Use component’s shape to filter out small pieces and noises, to get only long curves.



Step 2: Trimming the curves

- Since we want to generate a grayscale image in the future, we want to make sure that the curves are of same length.
- Iterate through the extracted curves to check the lengths, trim longer curves from tail to match the shortest curve.

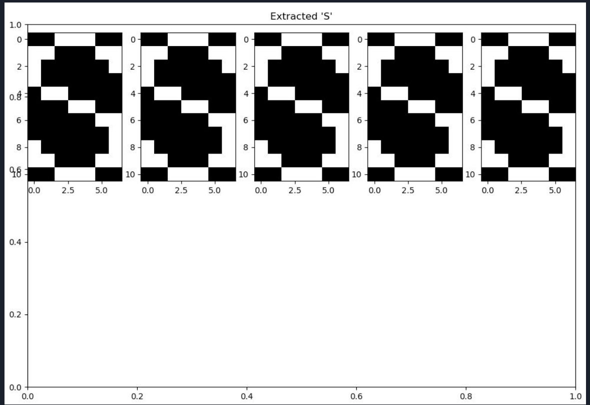


Step 3: Find the baseline for grayscale conversion

- We want to define a reference axis for each curve so that we can record the amplitude of each peak. The baselines need to be at the respectively same location for each curve so that they are on a same scale.
- For human eyes, it's easy to spot a reference axis just by looking at it. But for a computer algorithm, we need something more specific and accurate. In this case, we find all the 's' characters in the tag in front of each curve, using filtering and OCR, and define the lower border of 's' as the reference axis for each curve

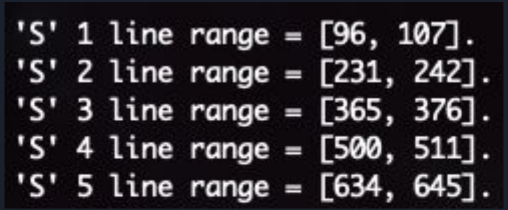
Tag

'S'

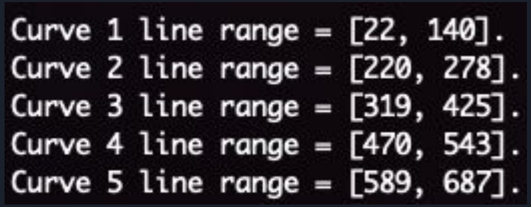


Step 3: Find the baseline for grayscale conversion

- Find all the ‘S’ in the original image, and find the range of the curves.
- The lower bound of ‘S’ would be the reference axis for the corresponding curve.
    - In this case, line 107, 242, 376, 511, 645 are the axes, and they all fall within the range of the curve.



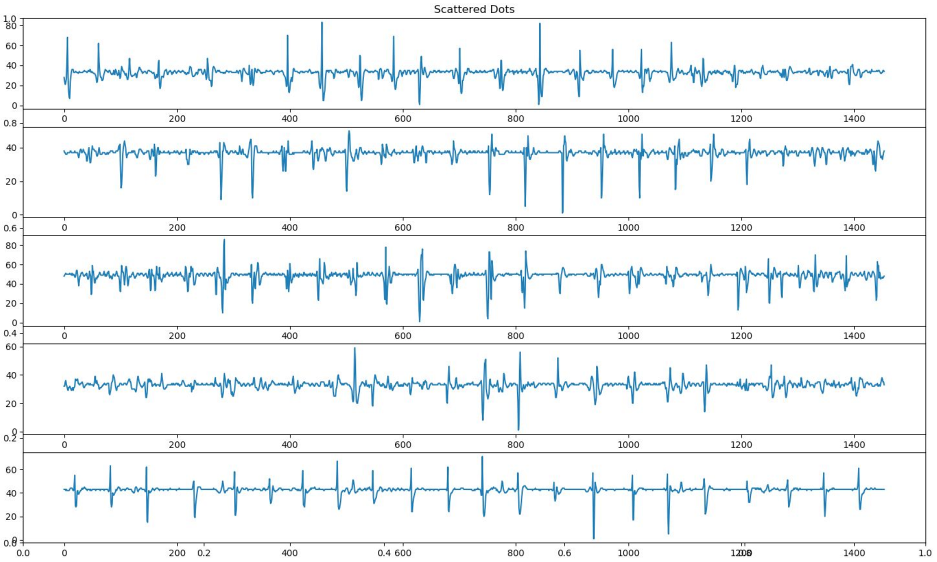
```
'S' 1 line range = [96, 107].
'S' 2 line range = [231, 242].
'S' 3 line range = [365, 376].
'S' 4 line range = [500, 511].
'S' 5 line range = [634, 645].
```

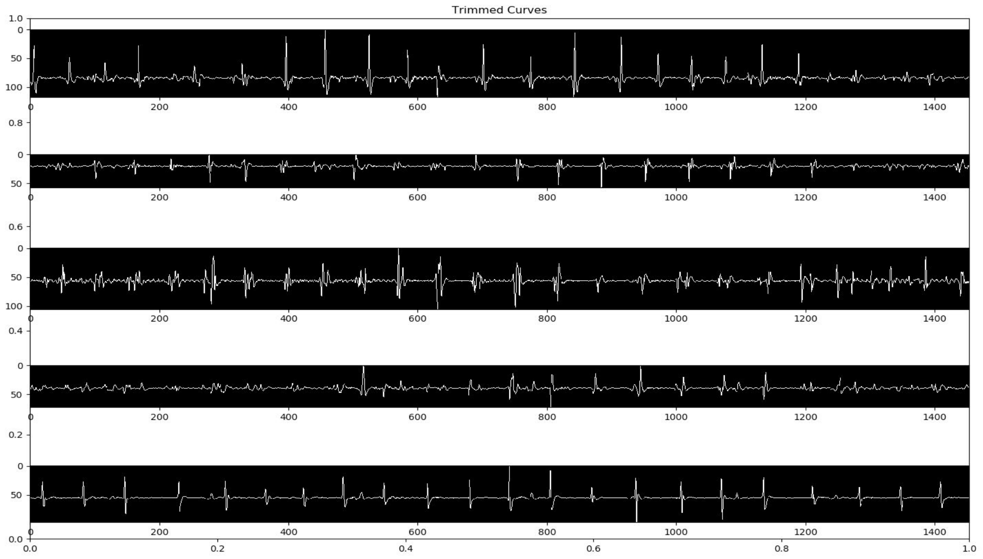


```
Curve 1 line range = [22, 140].
Curve 2 line range = [220, 278].
Curve 3 line range = [319, 425].
Curve 4 line range = [470, 543].
Curve 5 line range = [589, 687].
```

Step 4: Extract curve data from curve image

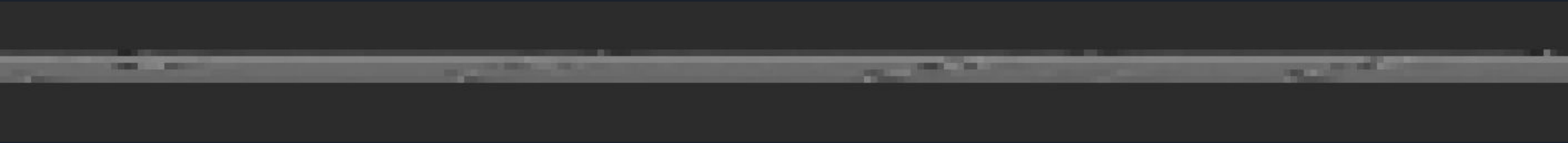
- We have the extracted curve from the raw image. We now want to extract the actual data each curve is containing.
- Scan through each curve, run a tracking algorithm that preserves the shape of the curve.
- We can see that it is almost identical to the original curve image.





Step 5: Generate the grayscale image

- Using the baselines we mentioned before, we are able to generate a grayscale image which contains the information of the current ECG. Below is a specific area on the grayscale image.
- The size of the image should be (5, length of curve), since we have 5 curves on a single ECG combined into 1 grayscale image.



Note: based on the size (width) of our curves, we designated the baseline to be grayscale == 127, whiter part means that it's a peak, and darker part means that it's a valley.

Future Work

- The precision of curve extraction on internal cavity ECG can still be improved.
- Find a better and less calculation-heavy way to denoise the scanned ECG.
- Find a way to do local dilation and erosion to avoid the situation where the label was merged with the curve.
- Scale up to a larger image dataset and add ML capabilities to make possible diagnoses on the generated grayscale image.