Tracking Software

Luis Nicolás Luarte Rodríguez

Main idea

- At each time step:
  - Separate background from foreground
  - Get the head point
  - Get the "body" point
  - Get the tail point
  - Do further analysis with those points

Ideal input

- First step is to have a good idea of what our background is
  - We can create a model (more, complex but better suited for environments with dynamic lighting)
  - Or we can assume that out background is never going to change (how most labs do it)

We can get an image of our background

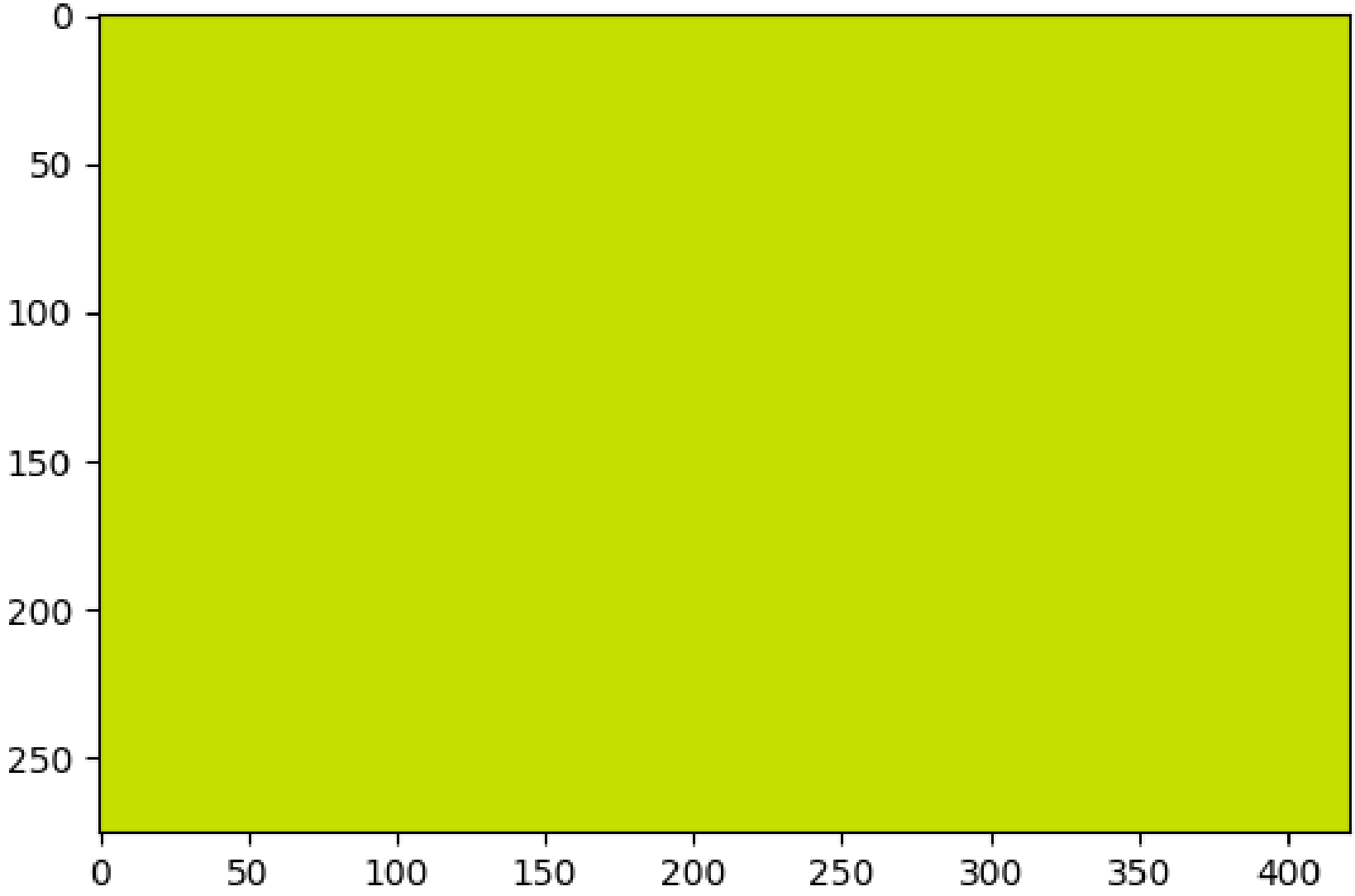


Figure 1: Example background

However a simple image is rather complex, because each pixel has 3 channels (red, green, blue)

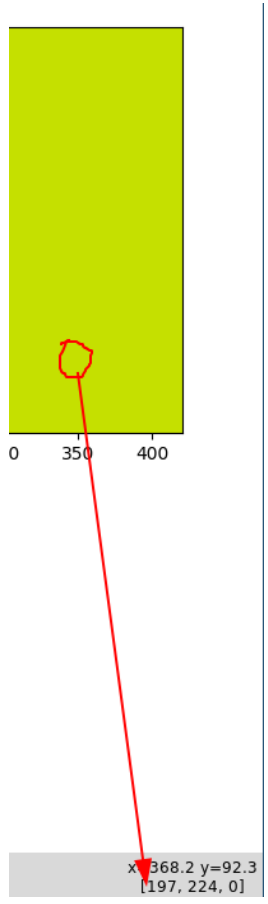


Figure 2: RGB channels

To simplify we turn it to gray scale (only 1 channel)

Figure 3: Grayscale = (r + g + b / 3)

However, we need to further process our background to make it 'smoother'

- We need to remove noise
  - Noise are 'details' that make an image harder to identify
  - In other words, denoising is equivalent to make an image more homogeneous, while preserving edges
  - A bilateral filter does such thing

Noise image

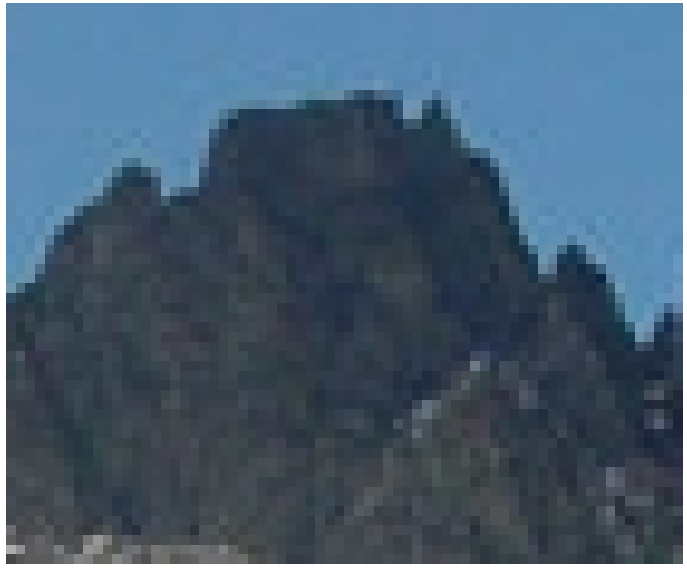


Figure 4: Notice the details

Noise image after bilateral filter

Figure 5: Details are gone, edges preserved

Similar steps are applied to the image with the animal

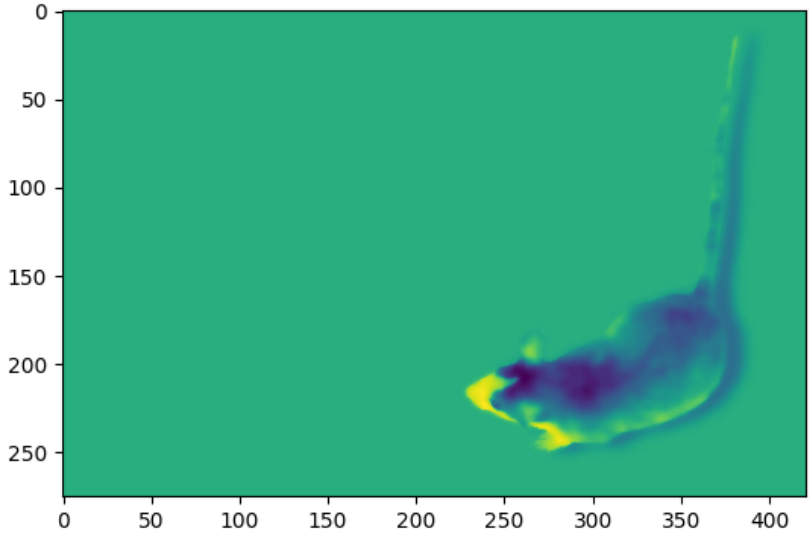


Figure 6: Smoothed image

The next step is to substract the background from the image with the animal

- This is done by substracting pixel intensity
- The result is not good, some areas of the animal are considered background (black color)

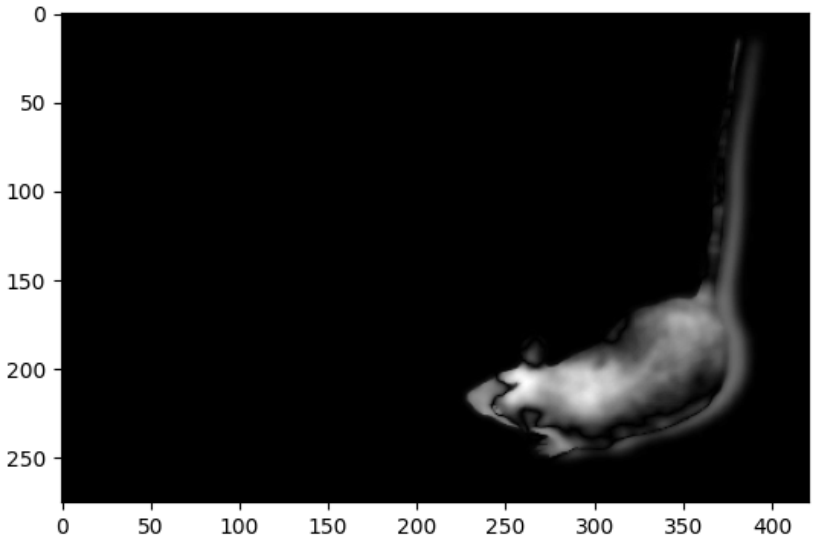
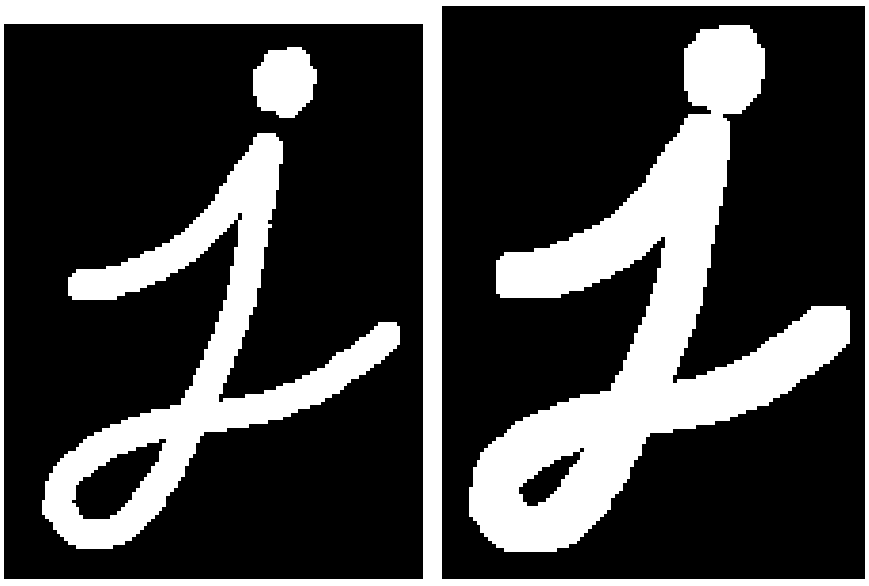


Figure 7: Difference

We use morphological transformations to fix this

- Morphological transformations are operations applied to binary images, which are based on the image shape
  - They use a 'kernel', which a windows where a certain operation is performed
- Our main problem is that there's background objects INSIDE the animal
  - The closing operation is applied to the kernel
    - A pixel element is defined as '1' if: inside the kernel there's atleast a '1' pixel (dilation)
    - Then we 'erode' the boundaries: a pixel is considered '1' is all pixels under the kernel are 1's, otherwise is 0

Dilation



Erosion

Closing = dilation followed by erosion

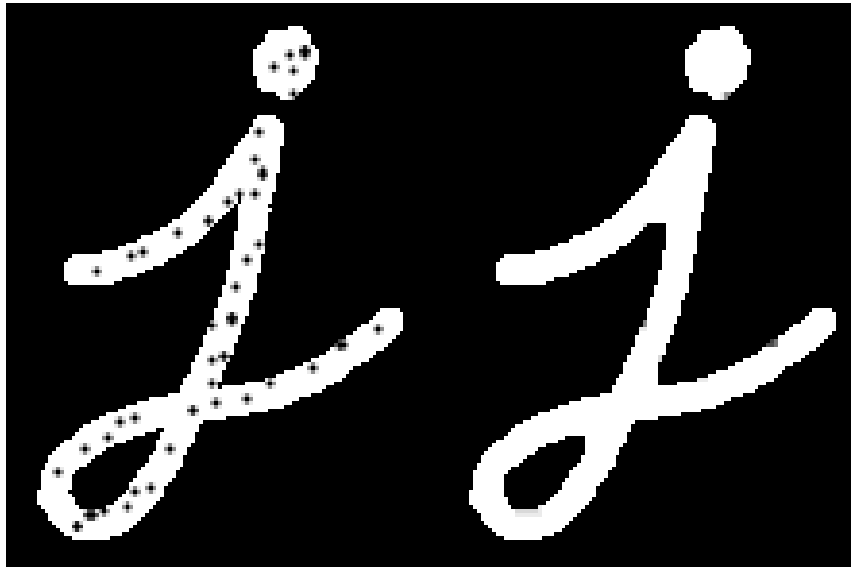


Figure 8: Closing

Result

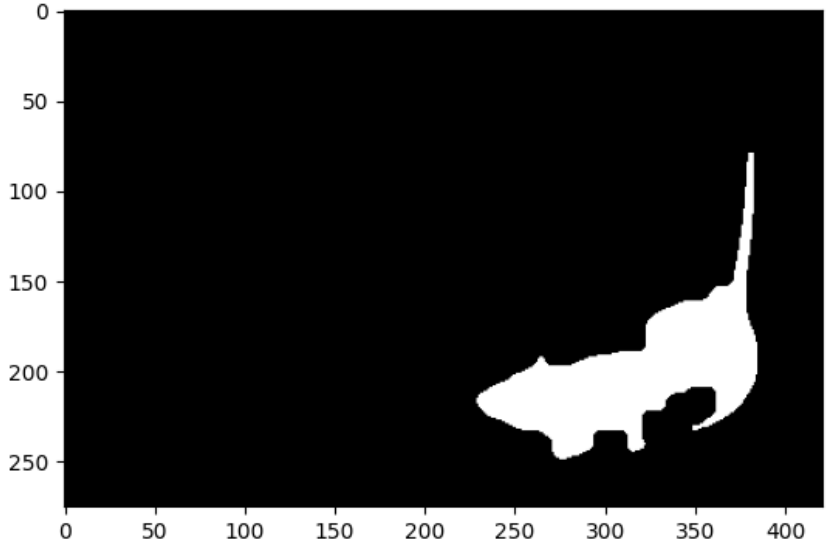


Figure 9: Not perfect, but the animal is clearly isolated

The original image is not ideal, but we can improve this and make it a more robust algorithm

- We can calculate the contours of the image
  - This makes it more robust against 'lighting artifacts'

Figure 10: Image with artifact

Artifact removal

- The algorithm simply calculates the contours of each objects, and selects the one with the bigger area
  - The image looks worse because the artifact is super big
  - In normal conditions an artifact like that can be removed manually

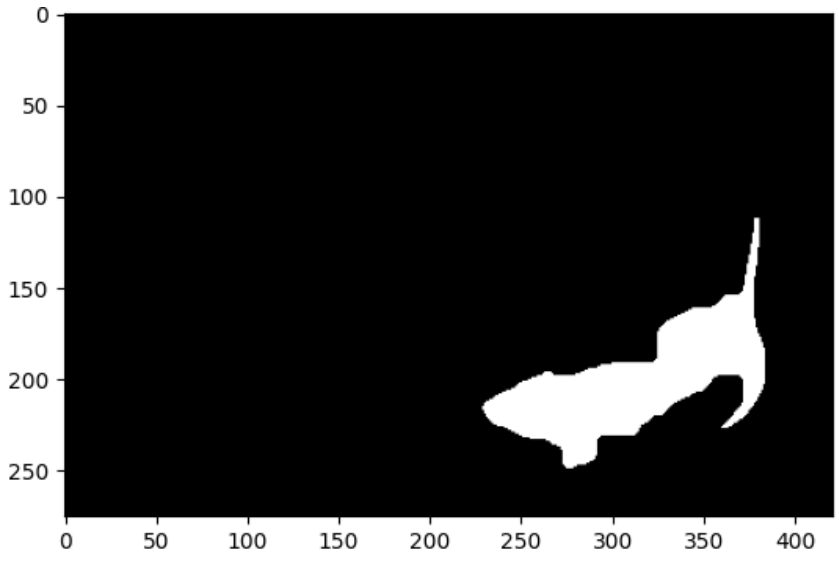


Figure 11: Image without artifact

After the segmentation problem, we need to find the head, body and tail

- The intuition is:
  - Rats/mice are ‘chubby’
  - Long tails
  - ‘small’ head relative to the body
  - The tail is the furthest away point from the body
  - The head is the furthest away point from the tail

The geodesic distance is the proper implementation to solve this

- The geodesic distance is a shortest path between 2 points in a certain space
  - We define our space as the animal surrounded by boundaries (background)
  - The distance, considering the boundaries, is the geodesic distance
  - Is approximated by the fast marching algorithm

Geodesic vs euclidean distance

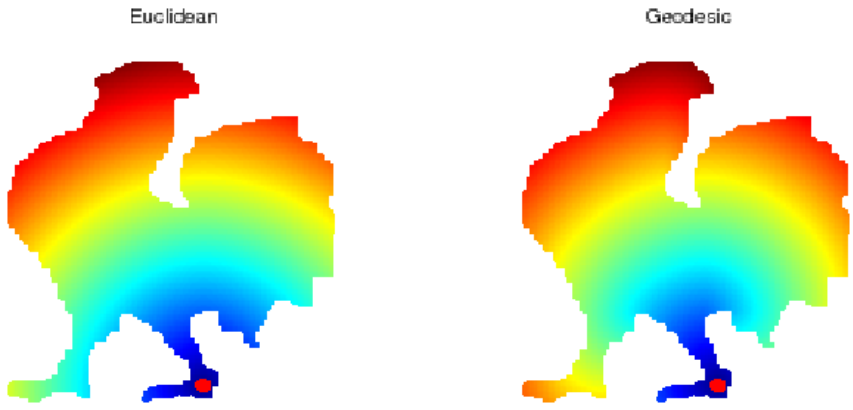


Figure 12: Notice how the boundaries inform the distance at the feet

Considering this, the furthest away point is the ‘body’ because the animal is ‘chubby’

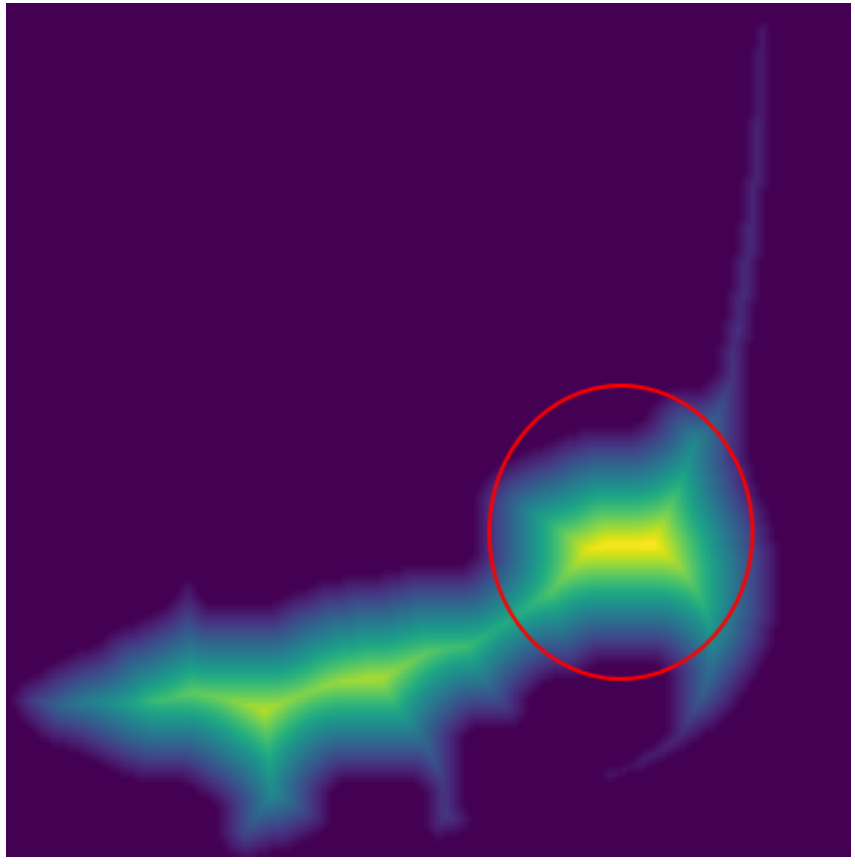


Figure 13: Warmer color are more distant

Detecting the head and the tail is relatively simple

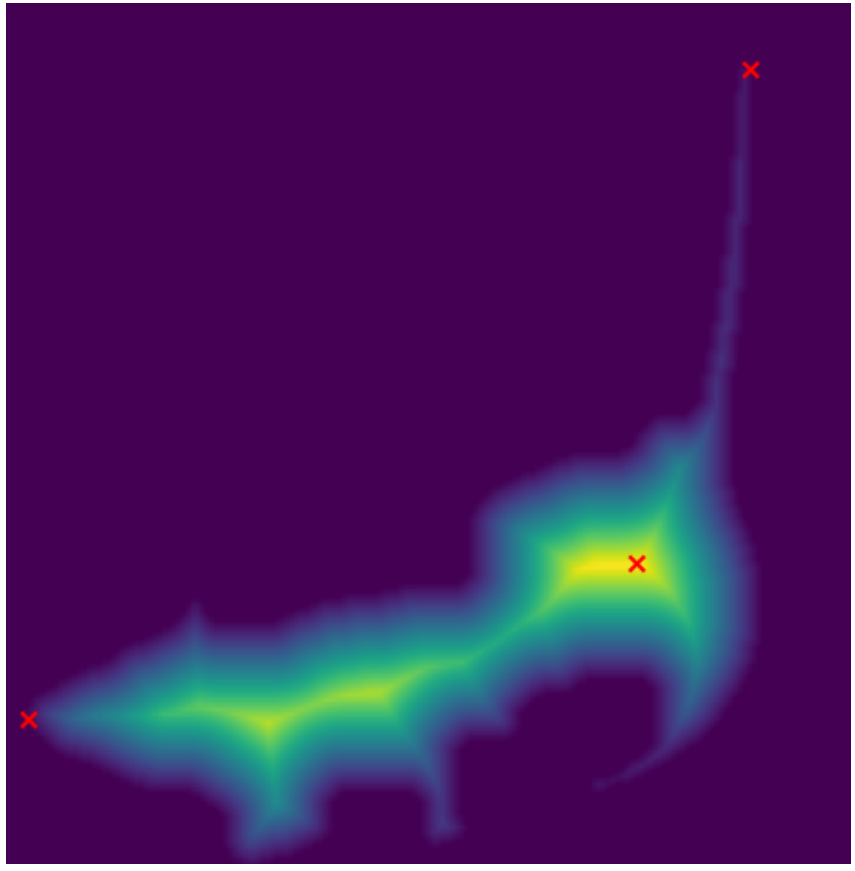


Figure 14: In read the calculated points

The final algorithm should be (not tested in real animals yet) resistant to rotations, minimal dynamics in lighting conditions and non-uniform backgrounds

show video

How to implement the system

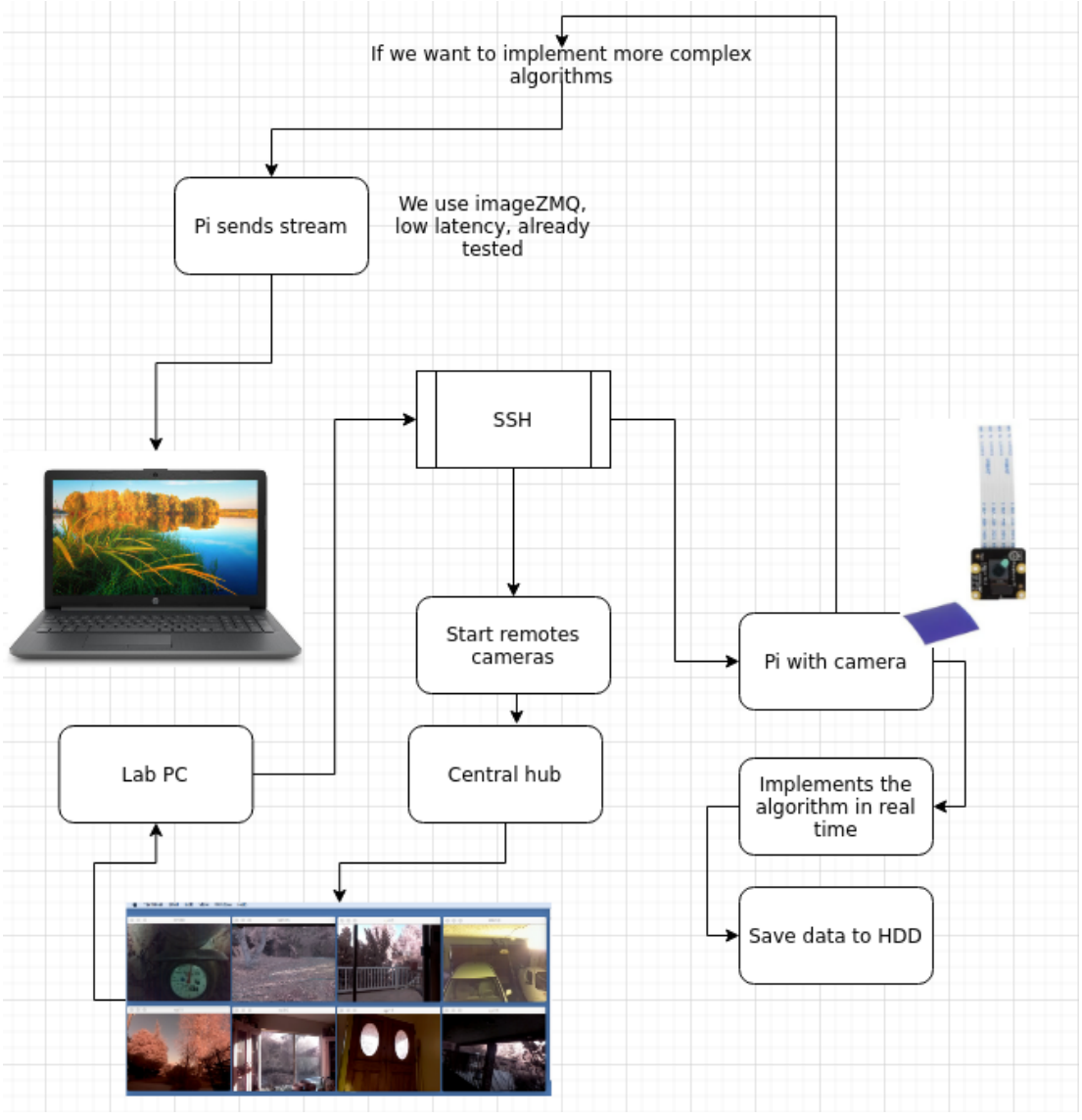


Figure 15: The big picture

TODO list

- Test with real animals
- Implement the GUI (only few functions right now)
- Test with multiple pi
- Stabilize frame-rate
- Pack and deploy into a docker (possibly)