Interactive Vision: Two methods for Interactive Edge detection.

Final project by Daniel Zatulovsky

Email: zatulovs@post.bgu.ac.il

Introduction:

The purpose of this project is to present a new research field, Interactive Computer Vision, convince the reader that such field is indeed useful and then discuss a specific problem in the field: Interactive edge detection, providing two distinct methods to approach it, each applicable under different situations.

Definition: Interactive Computer Vision is the field that tries to solve the Vision problem by getting limited help from a user and by so managing to greatly improve the results.

Applications:

There are numerous task that are hard to solve by fully automatic systems. Instead of trying to do so, we'll try to build a system that would require minimal user interaction, thus save effort and resources.

For example, let us think of the CROPS project. Obviously building a fully automatic system to harvest crops is a complicated task. What if, instead of solving it, we build a semi-automatic system with multiple cameras that photograph the field. There is a single worker on each field whose task is "clicking" on peppers he sees using a touch screen. The system uses this information to distinguish the peppers and send a robot to pick them.

Such solution isn't as far reaching as the autonomous robot, yet it still brings technological progress to the field and will save manpower.

Definition: Interactive edge detector is an interactive version of a typical edge detection algorithm which greatly improves on the performance of fully automatic systems by using/requiring simple, imprecise input from a user.

This technique is especially important on platforms where interaction with the user is fluid. Like smartphones and tablets.

A good example where algorithms such as these would be useful are mobile photo-related applications. A user could, inaccurately, mark the borders of an object and the application would use the input to recognize and process it. For example, you could accurately select and delete a person passing in front of the camera while the photo was taken (of course filling the blank would require a different, unrelated, algorithm).

Approach and Method:

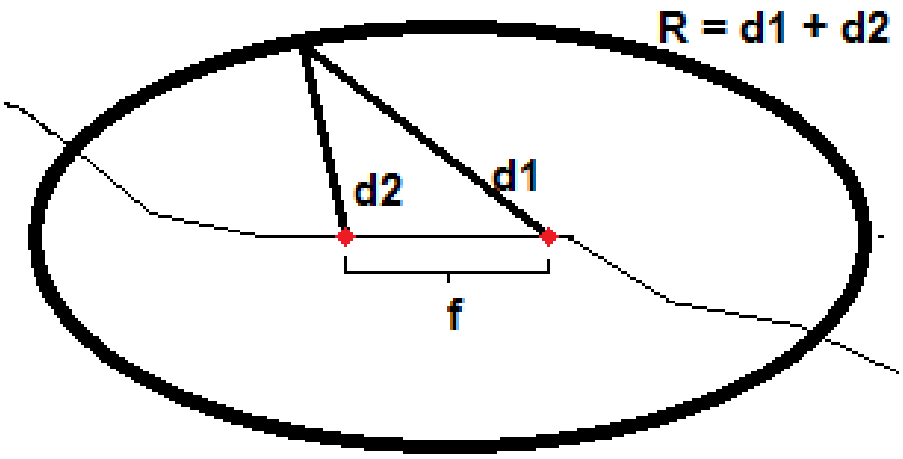
The first, proximity based, method requires the user to trace a curve around the object. The algorithm then decides whether an edge point should be kept or filtered based on its proximity to the curve.

For every point P of the input curve, an elliptic shape is built. The ellipse is centered around P, has the same approximate direction as the curve at P, has the focal distance *f* and the radius *R*. We then filter the original edge map of the image based on those ellipses.

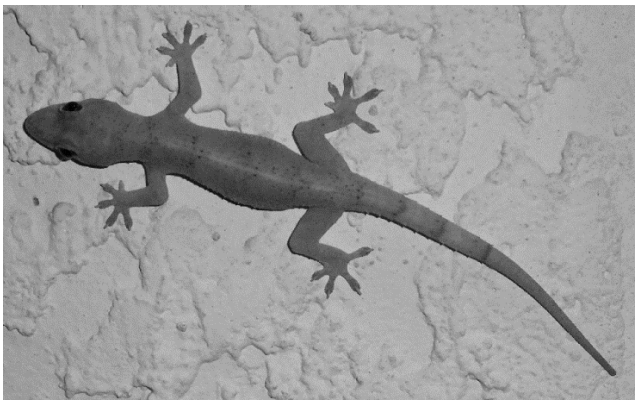
R is the main parameter, it directly decides the ratio between false positive and false negative detection.

Under constant *R*, a larger value of *f* means a thinner ellipse, which yields better results when processing straight lines, but is more prone to error when rapid changes in direction occur.

In the case that the direction of the curve couldn't be decided, a symmetric circle is used instead (*f*=0).

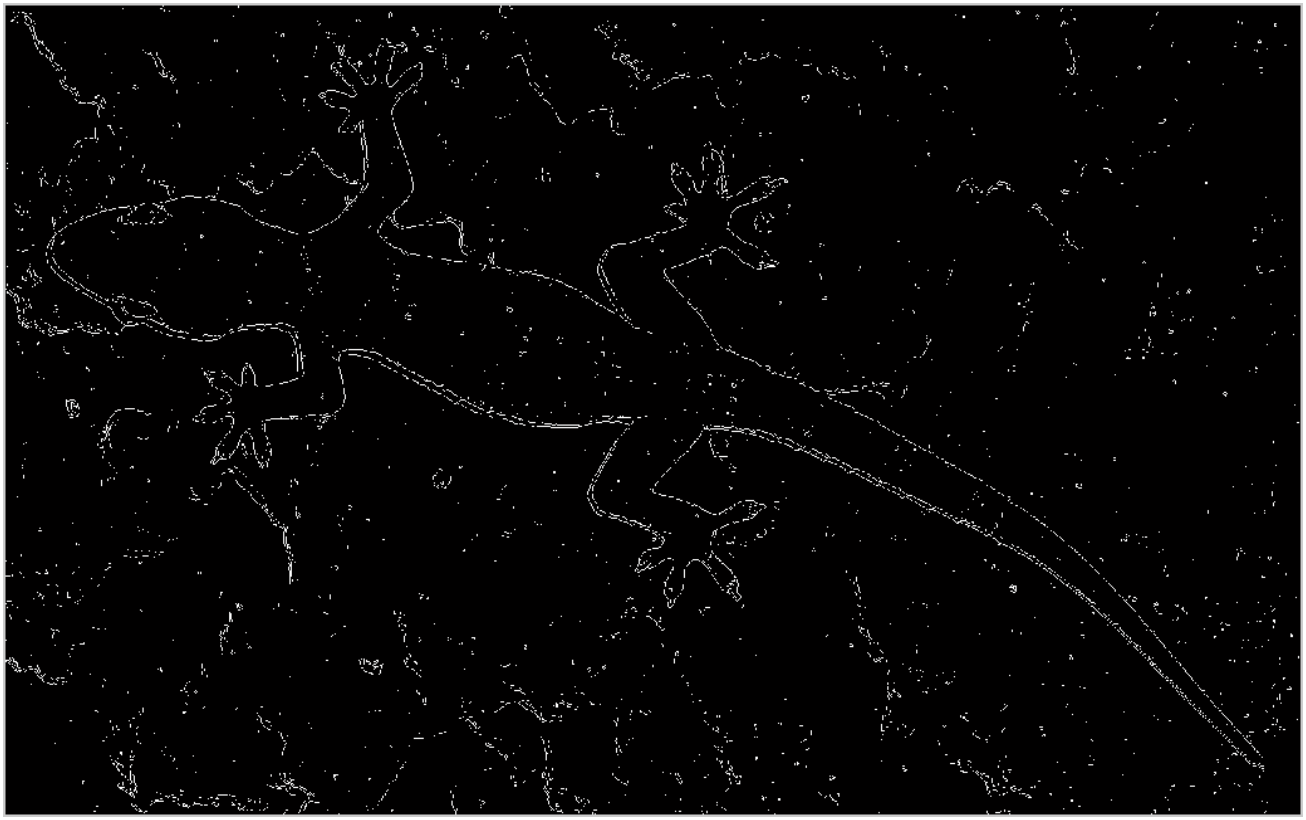


We'll have a consistent example though this file with the following pictures:

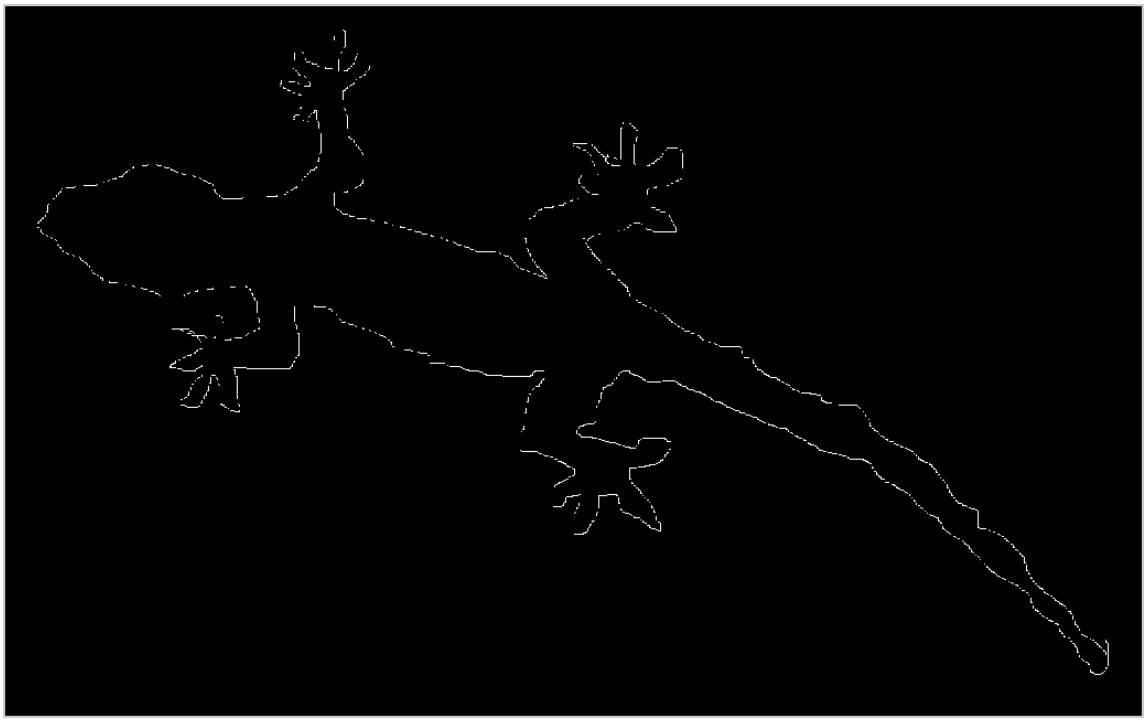


Results:

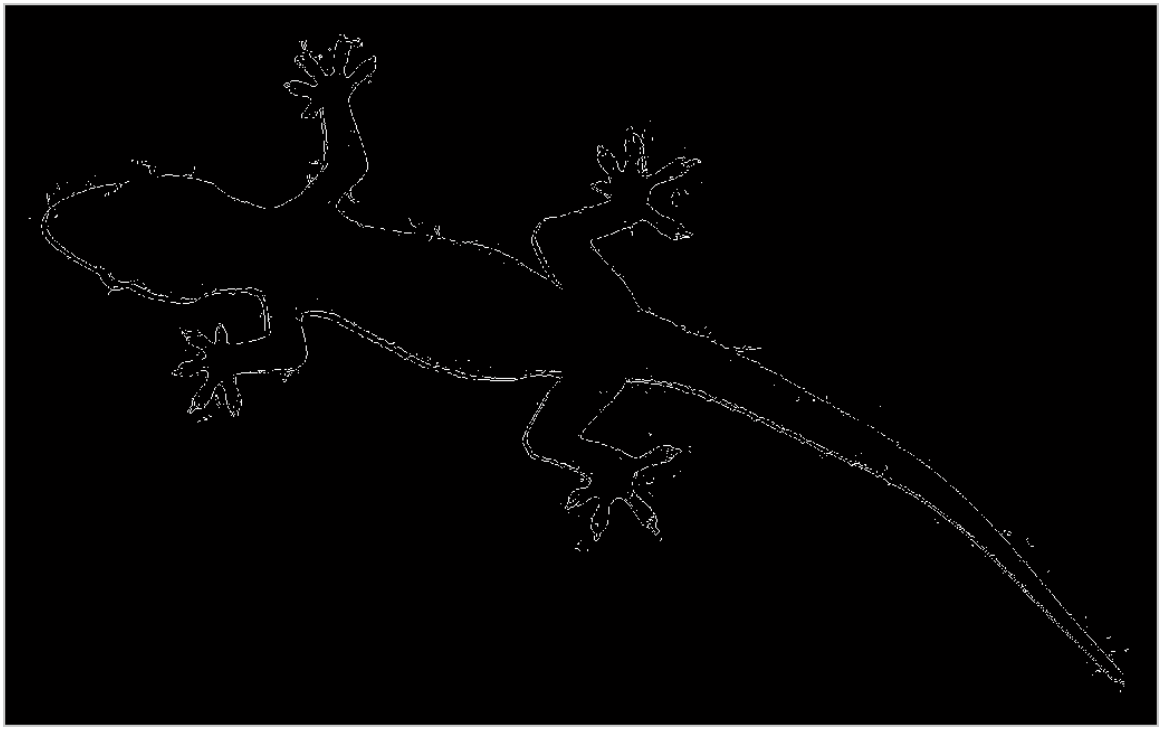
Let's look at the following noisy edge map:



By using the following input from the user

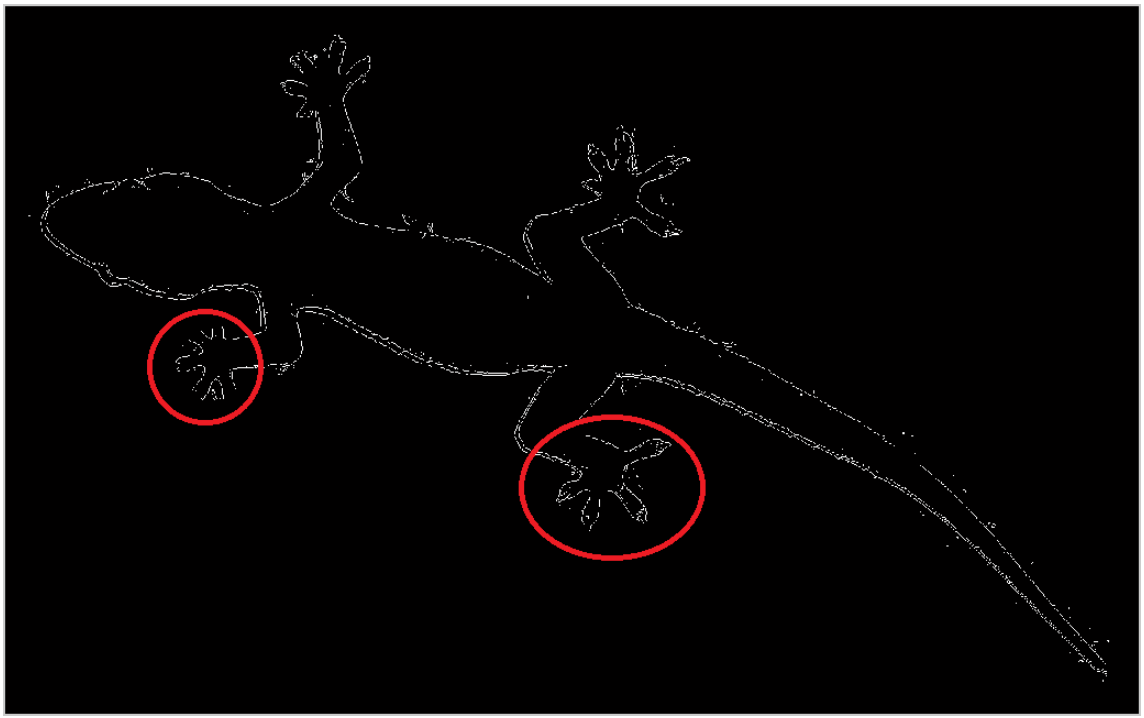


We achieved the following results:



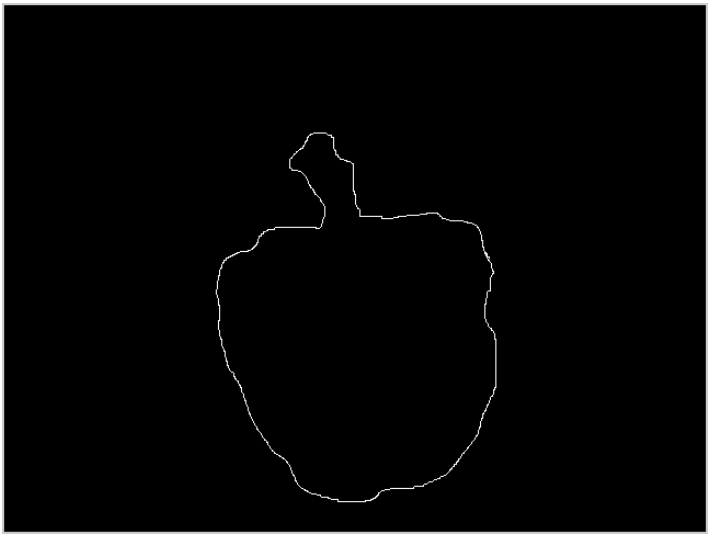
Let's look at harder case, where the input from the user is much less precise.

Produces the following output:

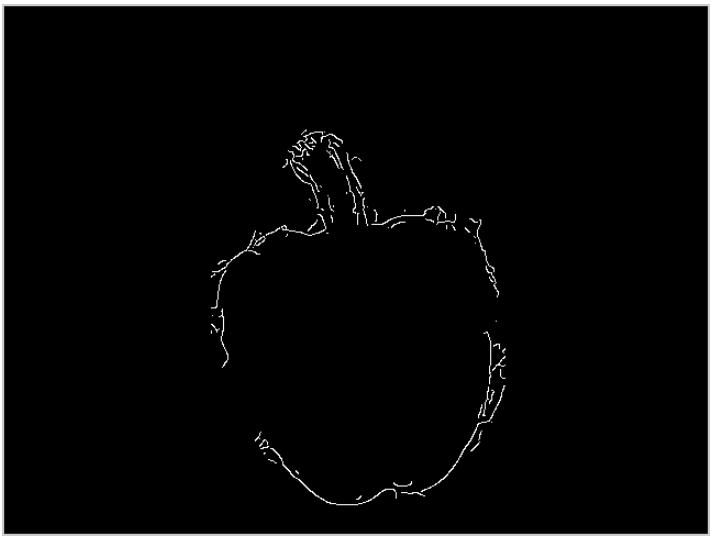


Another example: Combining the following edge map and input

+

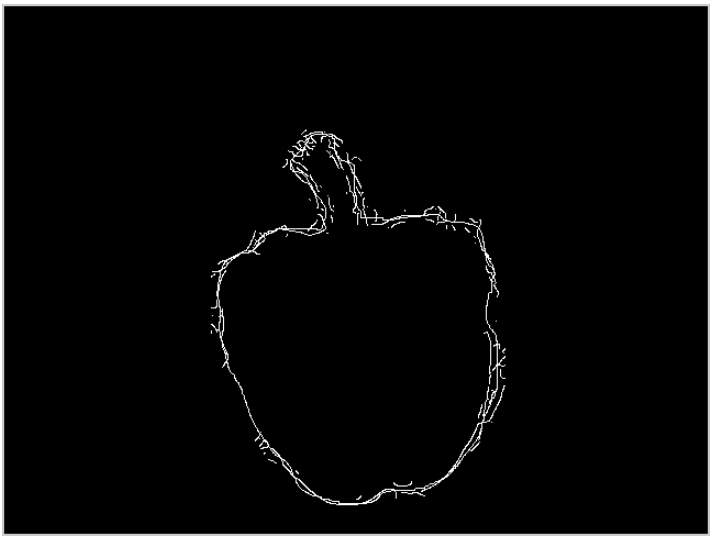


Will result in:

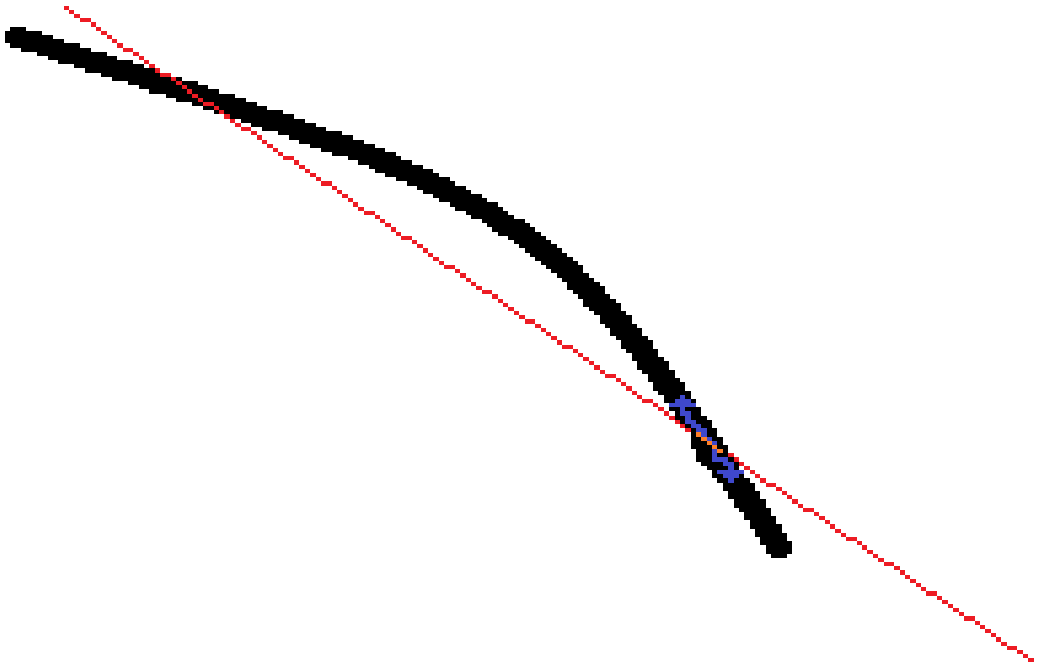


Not a perfect match this time, but we can farther improve it with heuristics involving the input.

An example for (very) simple heuristic would be looking at the union of the output with user's input.



The second method requires less precise drawing and uses only the intersection points between the user input and the edge map instead. This allows the user to use straight lines instead of curves.



Points in the intersection are marked as edges. Then a process similar to Canny's Hysteresis is used to mark all their adjacent points as edges recursively.

A window centered around each edge pixel is considered and all the pixels in it are marked as edges.

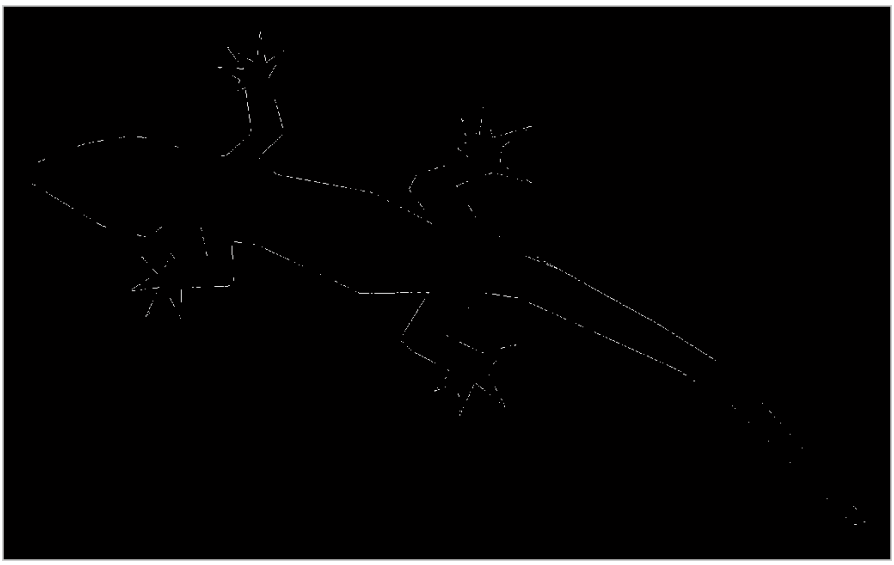
The size of the window is a parameter accepted by the algorithm. Larger window give better results on fragmented edge maps while smaller ones help filter noise.

Results:

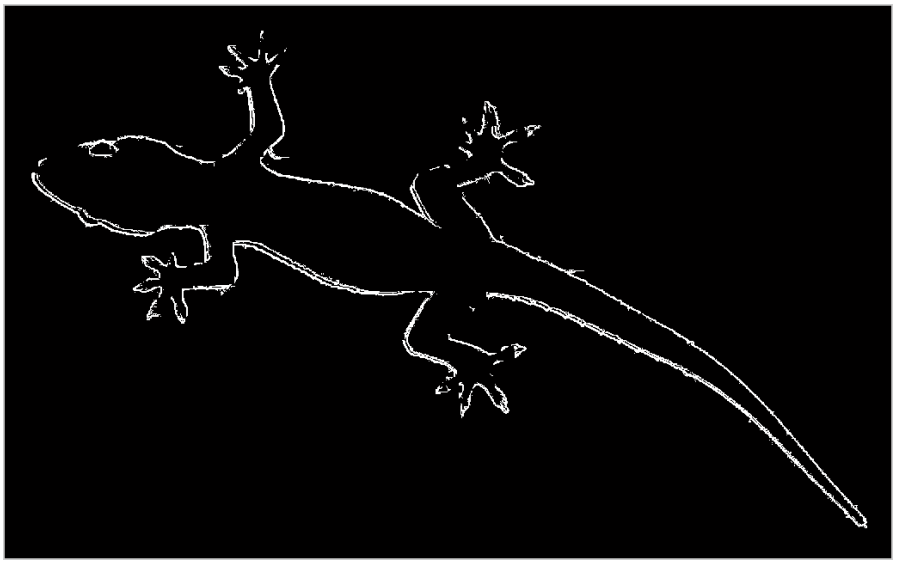
We'll use the following edge map. Notice how bright some of the edges are. This is important.

This is the input from the user, but we won't be using all of it.

We'll use only the intersection points with the above edge map.

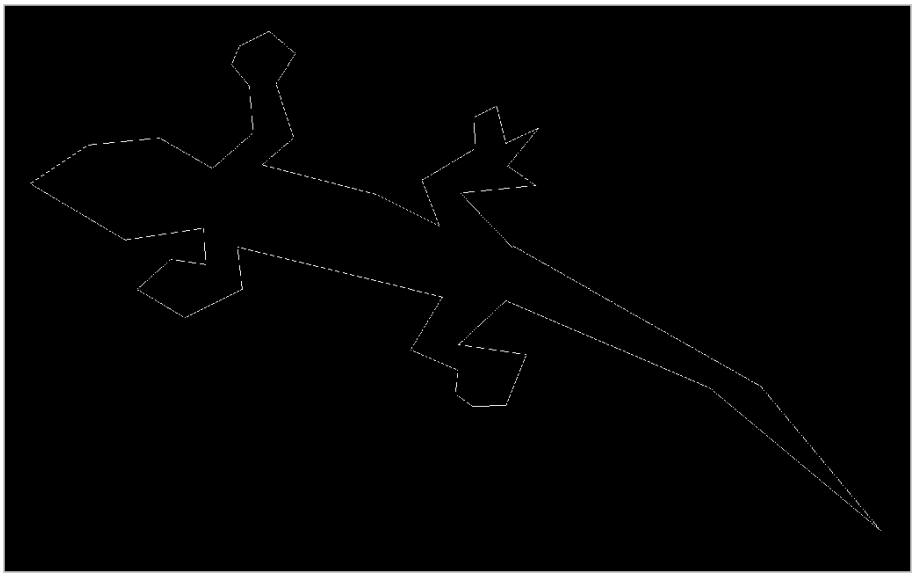


After the Hysteresis we'll get:

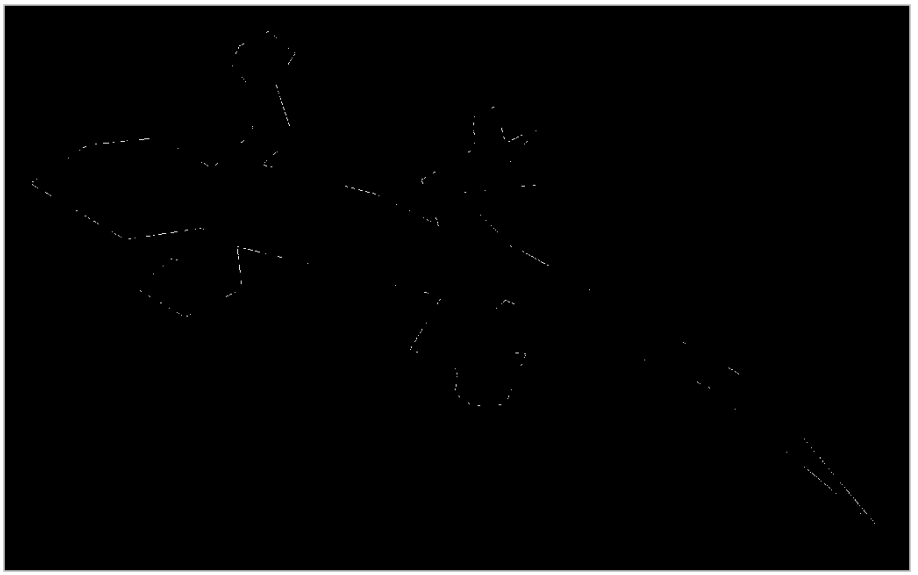


Another example, on rougher sketch.

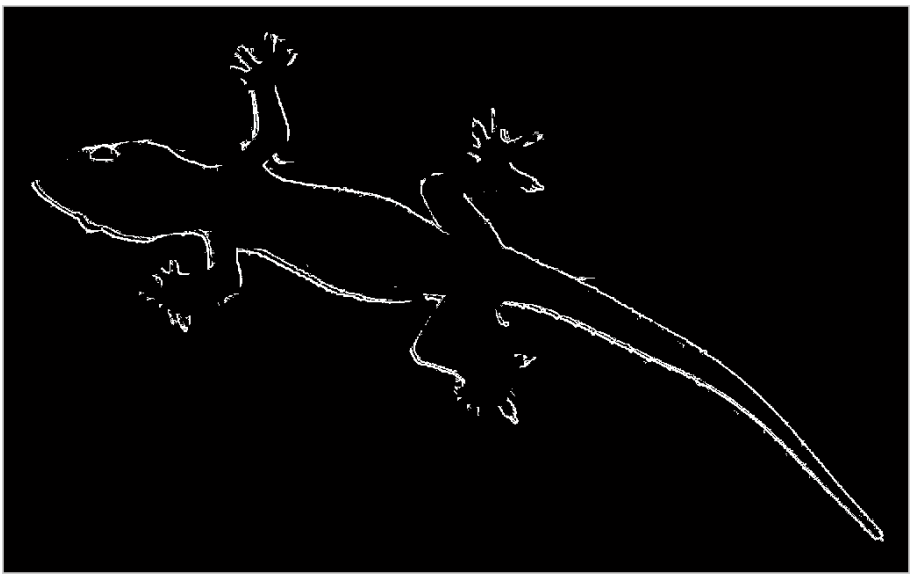
User Input:



Intersections:



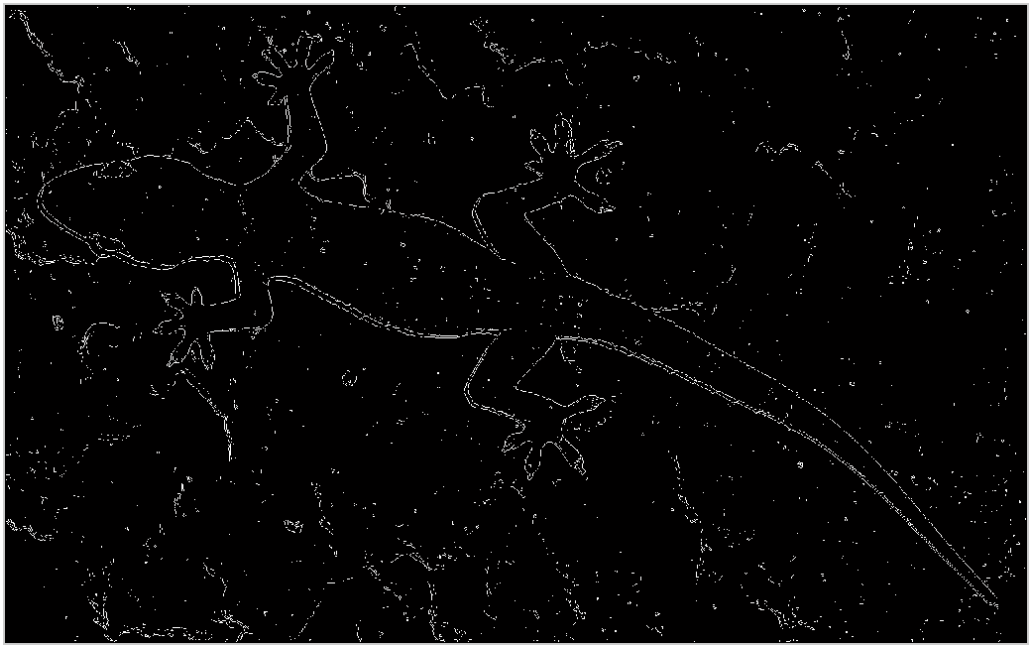
Output:



As stated before, this method relies heavily on the continuity of curves in the original edge map. Let's look at results from a different edge map.

Notice how similar this edge map is to the former one.

Yet the results now far less satisfactory.



Conclusions:

Both methods require as input both curve data from the user and an edge map.

The proximity-based method demands (relative) precision from the former, in return for lesser dependency on the latter, while the intersection-based method has the opposite dependencies.

Both provide effective tool to filter out unneeded edges. Whether caused by noise or unwanted objects. Furthermore because the results are highly independent, both methods can be used together. For example we could use the proximity filter to increase the number of intersection points or limit the wild spread of the Hysteresis.

I believe the results shown a sufficient reason for additional research on interactive vision techniques.

References:

[1] Canny Edge Detection

[2] Canny OpenCV

[3] Geometric properties of ellipses

[4] Efficient Queue in Matlab

[5] Introduction to Computational and Biological Vision

[6] CROPS homepage