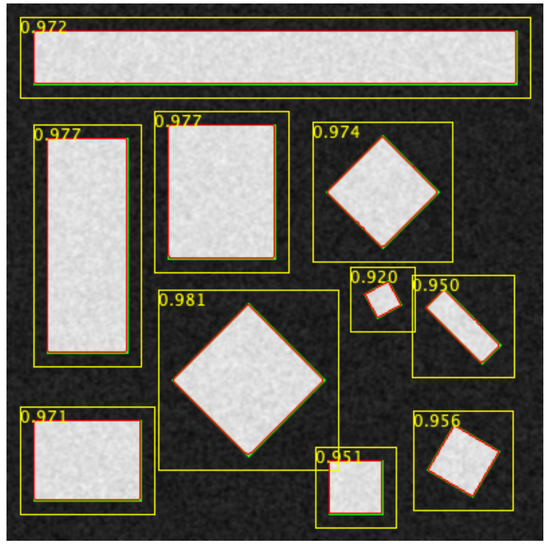
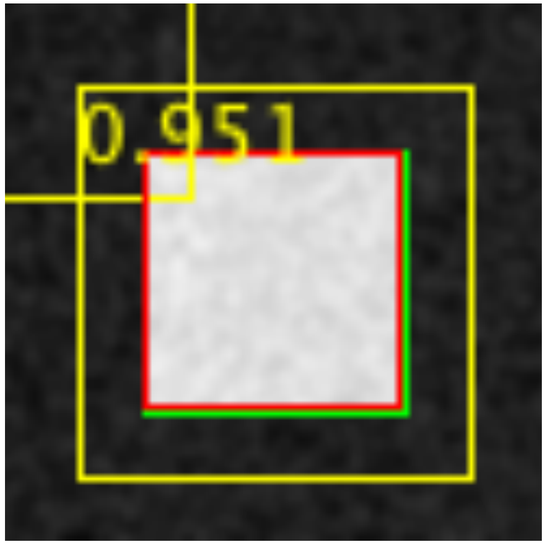
Problem Shrinkage Down-Right

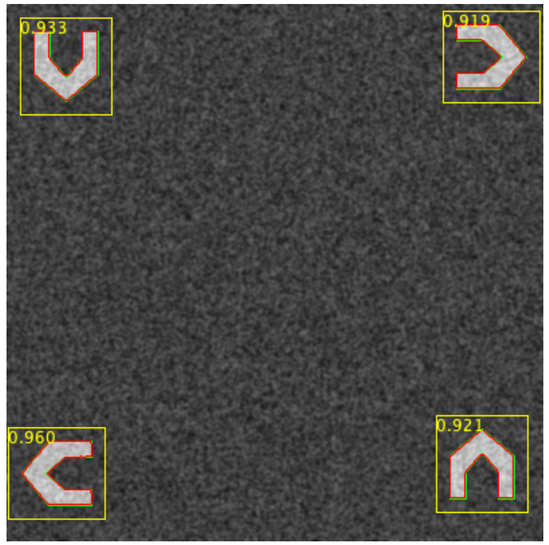
Prompt Ground-truth SAMJ

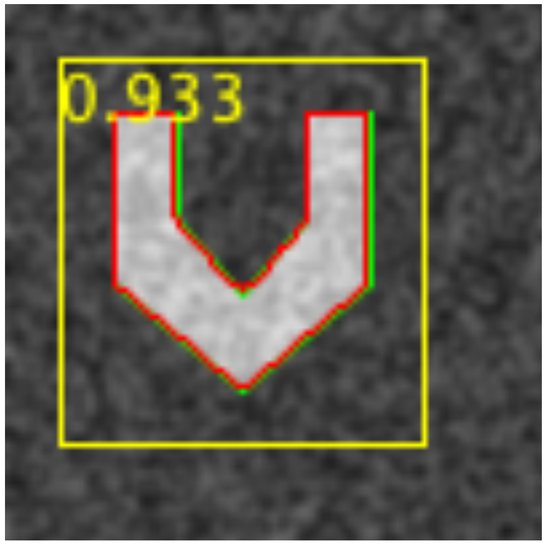
Observation

- for all models
- for all prompts





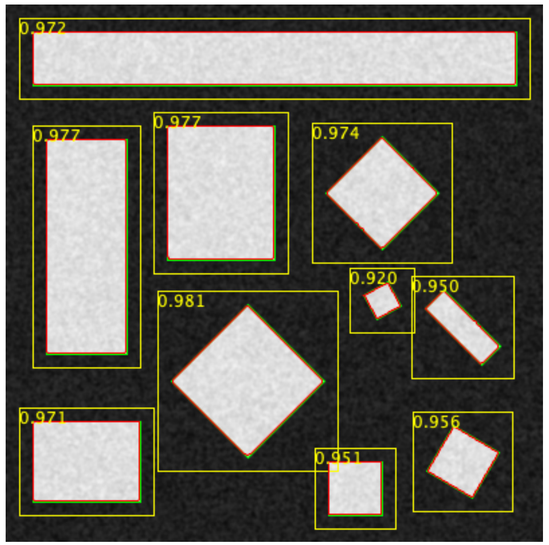




To be fixed

Problem Permutation X-Y Coordinates

Prompt Ground-truth SAMJ



Required a X Y permutation

- to be compatible with ImageJ

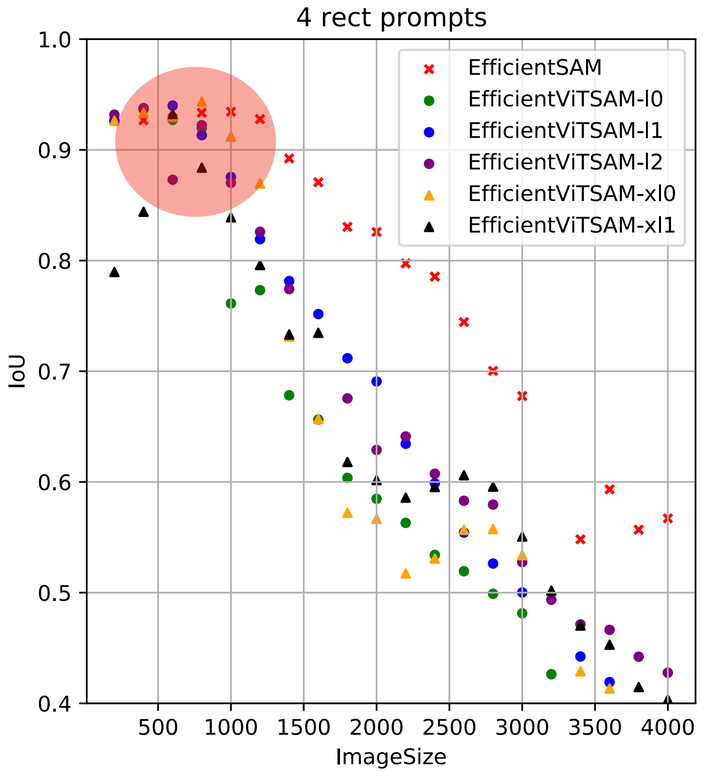
```
Rectangle prompt = region.rectangle(margin);
prompt = Tools.permuteXY(prompt); // TODO Permutation Axis XY
Region sam = instance.annotate(prompt);
sam.permuteXY(); // TODO Permutation Axis XY
prompt = Tools.permuteXY(prompt); // TODO Permutation Axis XY
overlay.add(new Roi(prompt));
overlay.add(region.getRoi(Color.GREEN));
overlay.add(sam.getRoi(Color.RED));
```

```
public void permuteXY() {
  for (int i = 0; i < npoints; i++) {
    int temp = ypoints[i];
    ypoints[i] = xpoints[i];
    xpoints[i] = temp;
  }
}
```

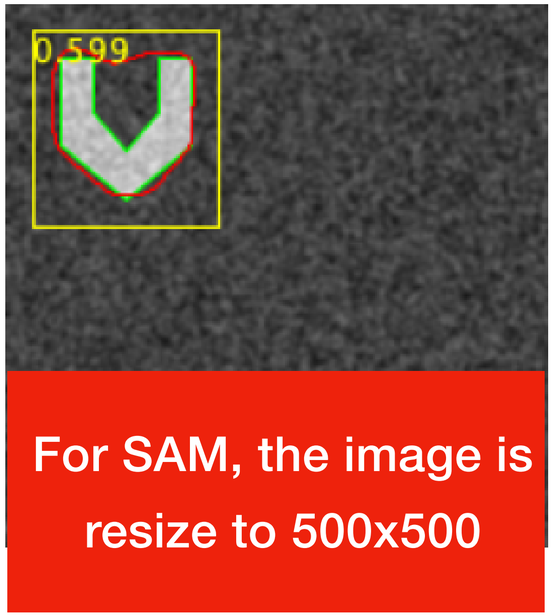
Scaling Effects of SAM

Experiments

- MacOS 13.15, Intel Chip
- Image size from 200x200 to 4000x4000
- 4 objects of the same size

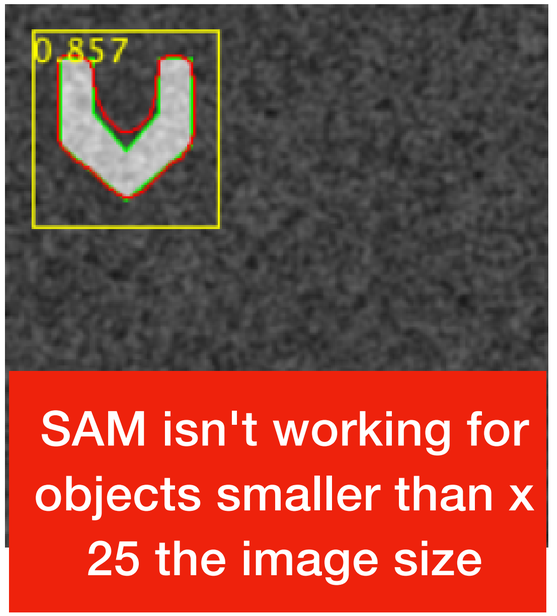


200x200 zoom up-left corner of image 4000 x 4000



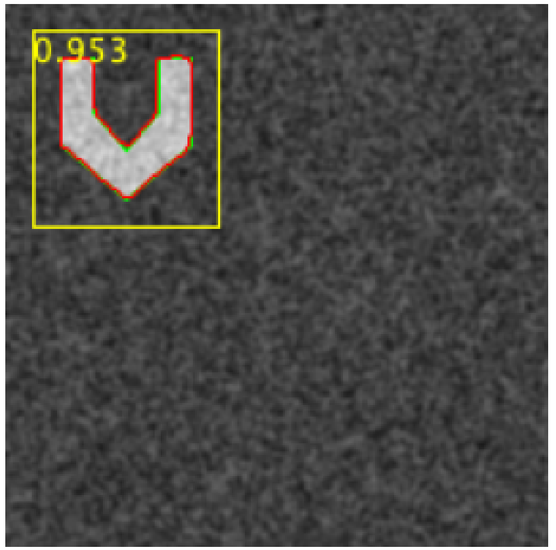
For SAM, the image is resize to 500x500

200x200 zoom up-left corner of image 2000 x 2000

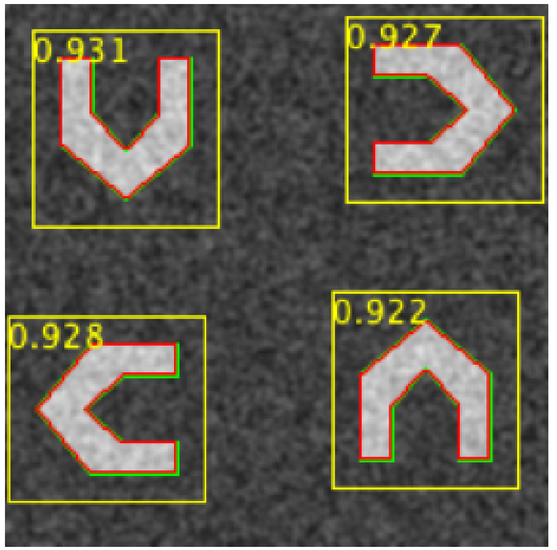


SAM isn't working for objects smaller than x 25 the image size

200x200 zoom up-left corner of image 1000 x 1000



200x200 zoom up-left corner of image 200 x 200



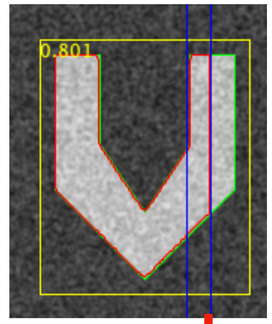
Tiling Strategy Implemented in Java

Tiling on the Java side

- MacOS 13.15, Intel Chip
- Image size from 2000x400
- Tiling by the minimum number of patches 512x512 to cover the whole image

Cut on the edges

Creating larger overlaps is too computationally heavy



Prompt

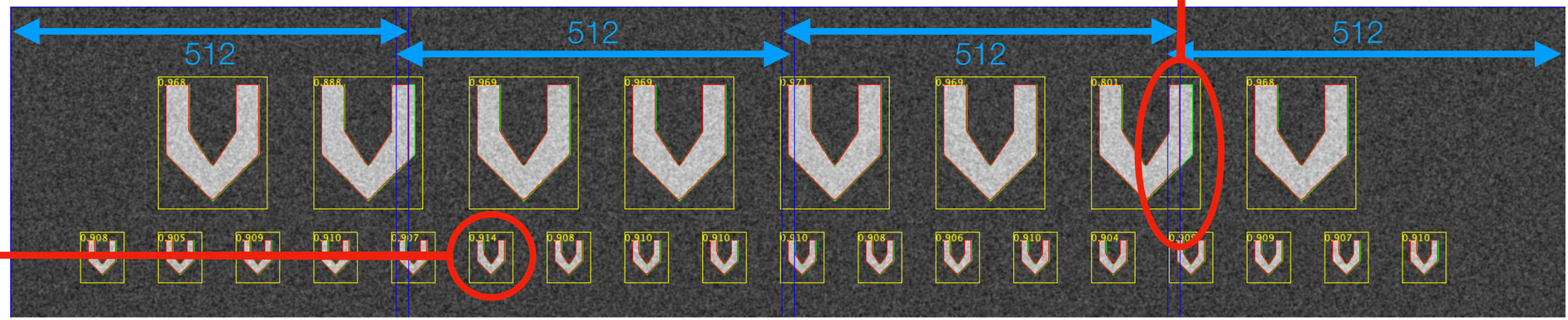
Ground-truth

SAMJ

Tile border

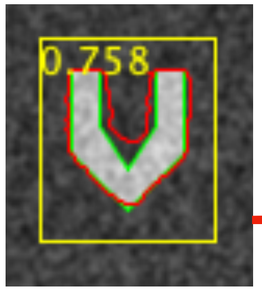
Tiling

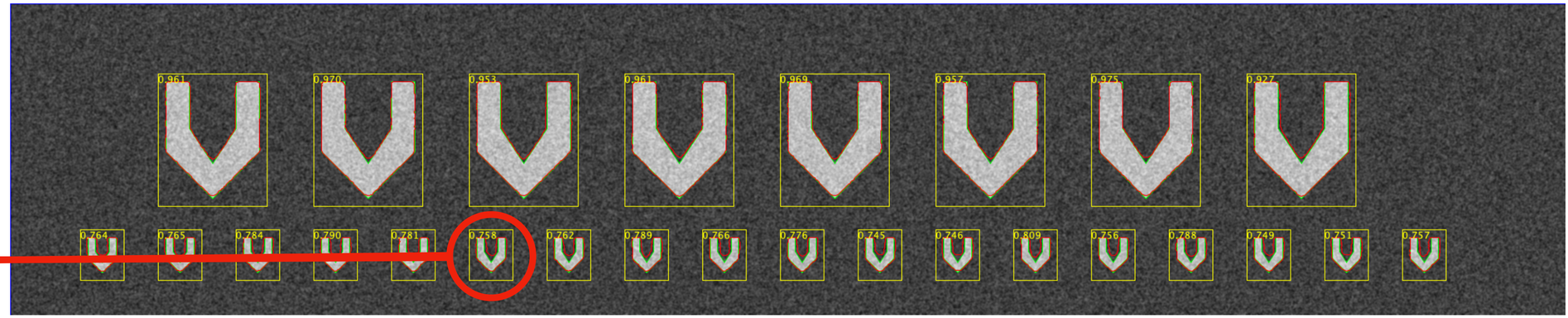
4 tiles
30 sec.
8 Gb RAM



Downsampling

1 tile
7 sec.
1.7 Gb





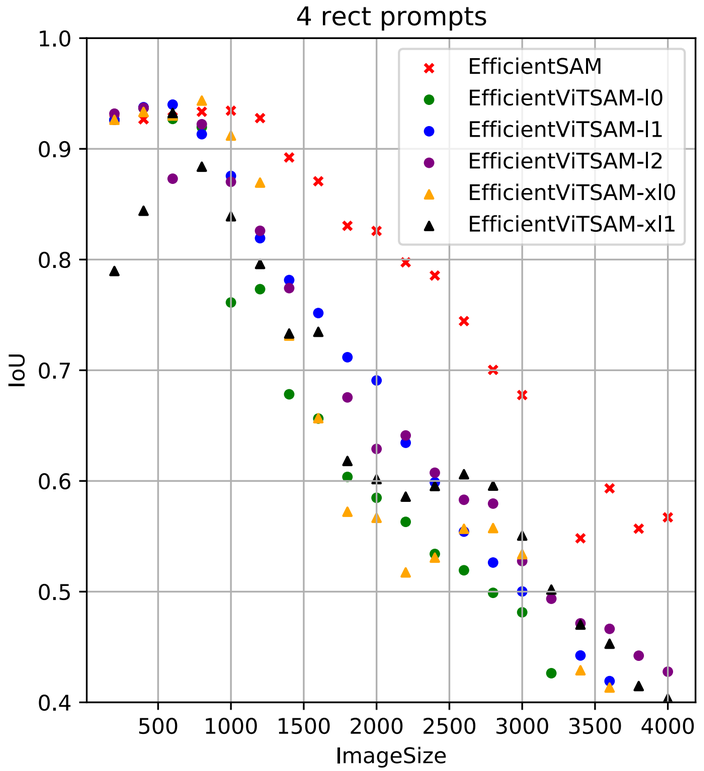
Encoding Timings

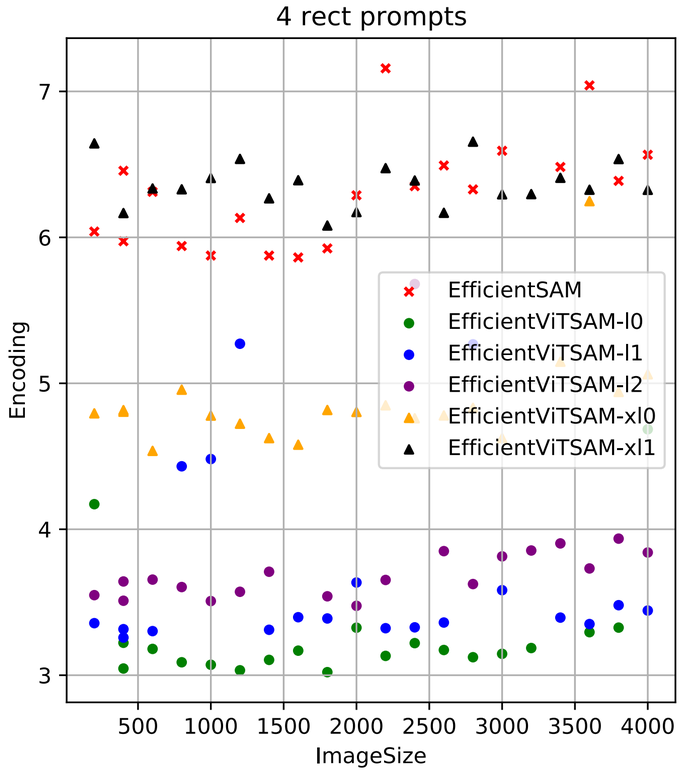
Experiments

- MacOS 13.15, Intel Chip
- Image size from 200x200 to 4000x4000
- 4 objects of the same size

Encoding time does not depend of the image size

Keep only EffSAM and EffSAM-ViT-l1





EffSAM and ViT-xl1 have the same encoding time but the quality of segmentation of EffSAM is better

ViT-xl0 is slower than l0, l1, l2 with equivalent results

l0, l1, l2 are almost the same in time and in quality

Annotation Timing

Experiments

- MacOS 13.15, Intel Chip
- Image size from 400x400
- 10 objects of various sizes

Annotation time

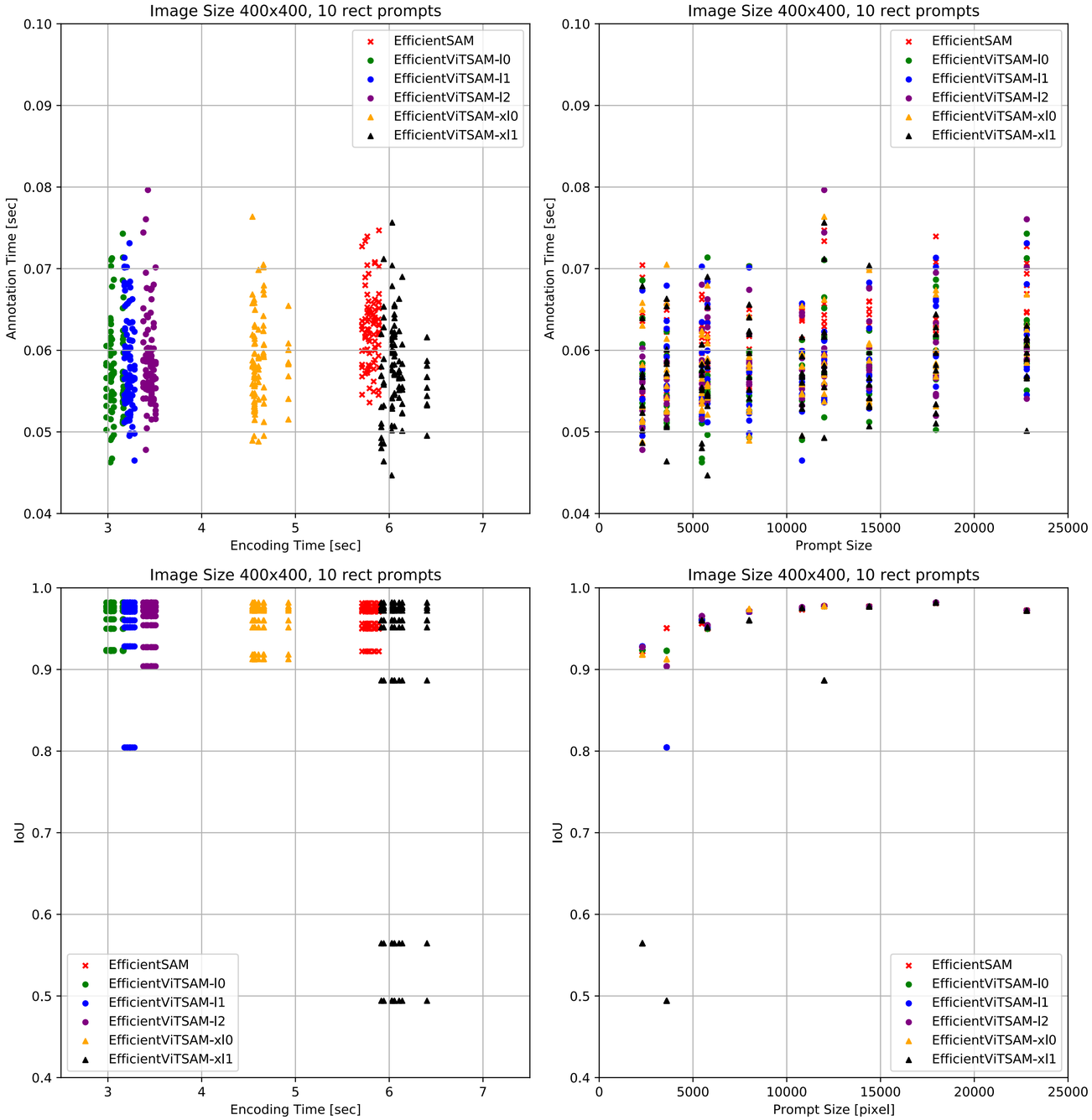
- highly variable from 40ms to 80ms
- not depend of the prompt size
- is the same for all models

Annotation precision

- is poor for small objects

First Annotation is always slow

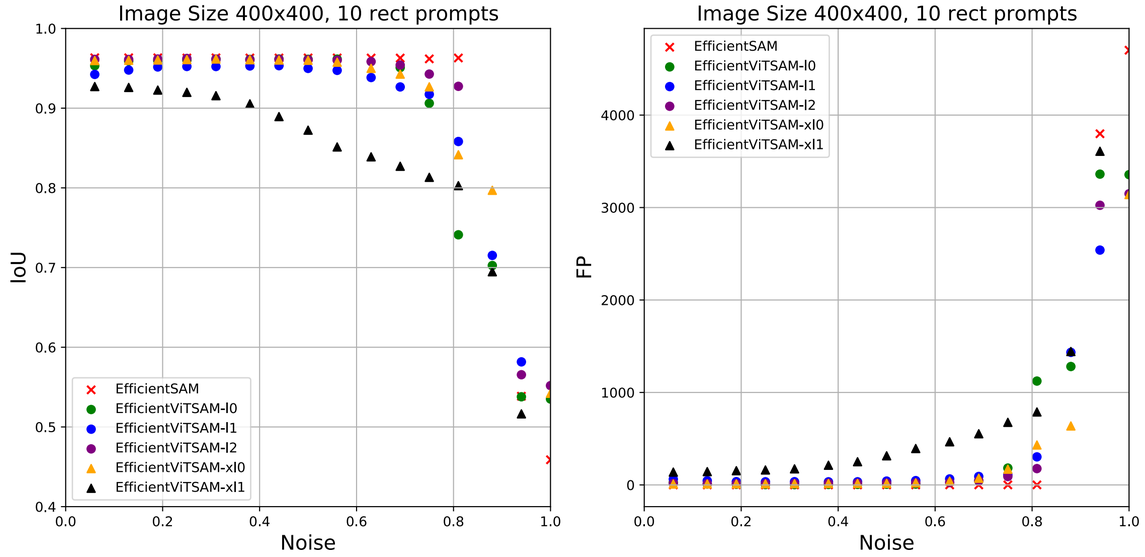
> Add a dummy test call of annotate right after the encoding without showing any results



Robustness to Structural Noise

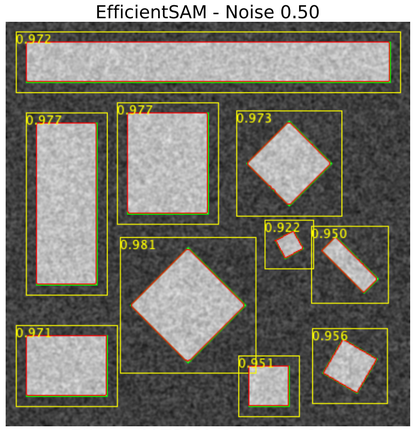
Experience

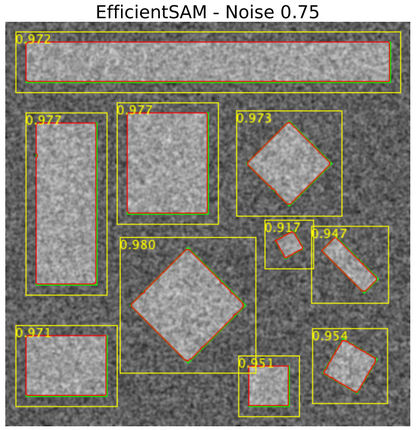
- image 400x400, 10 rectangular prompts
- image = (1-α) . signal + α uniform_noise

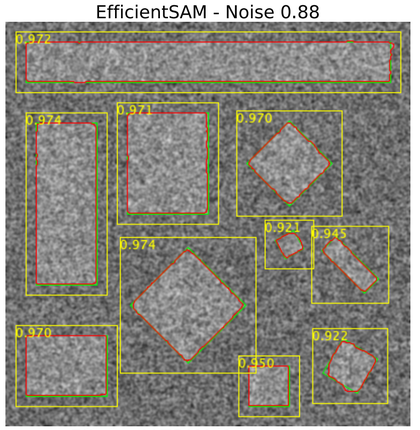


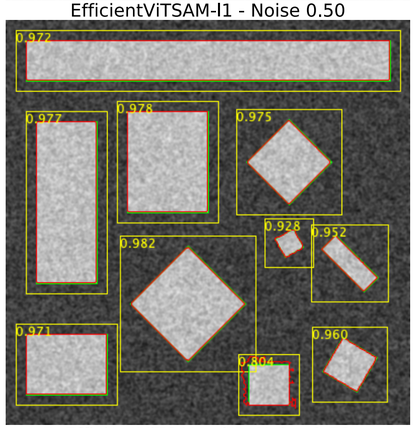
The two best ones: EffSAM and ViT-l2, ViT-l1

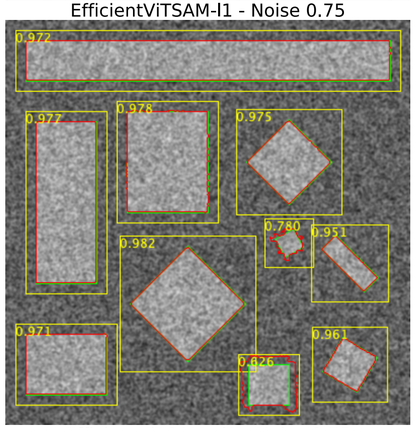
OK until α = 0.8

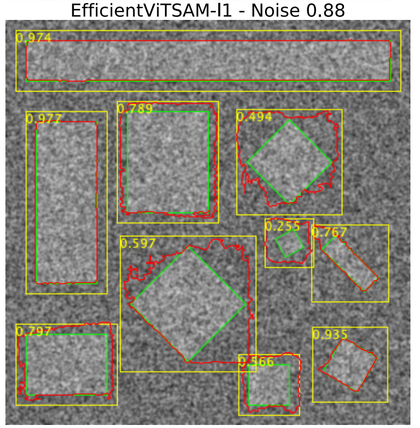


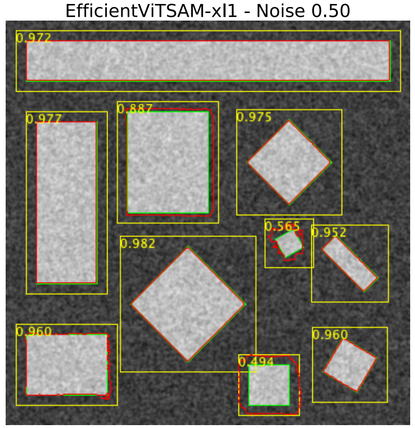


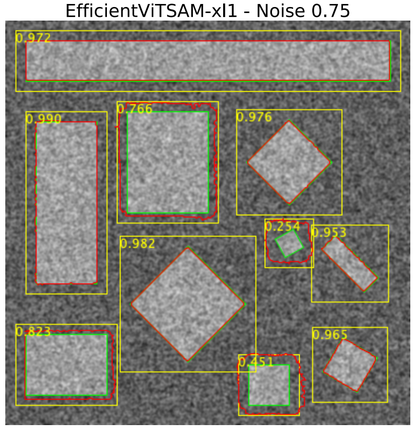


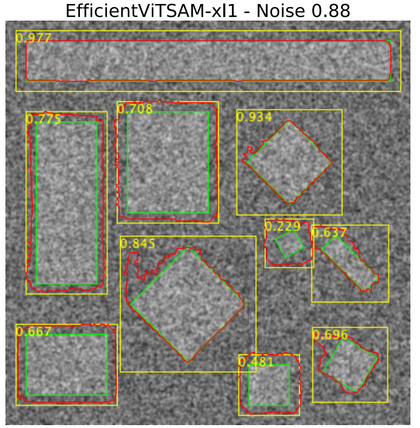












Coding

SAMJ sand-box

- Client ImageJ or Icy, in the future we could have QuPath, BDV

Debug mode ?

- Log are scary!
- How to make a debug messages on demand

Python files as Resources files ?

```
  protected static String TRACE_EDGES = ""
      + "def is_edge_pixel(image, cx,cy):" + System.lineSeparator()
      + "    # assuming image[cy,cx] != 0" + System.lineSeparator()
      + "    h,w = image.shape" + System.lineSeparator()
      + "    if cx < 0 or cx >= w or cy < 0 or cy >= h:" + System.lineSeparator()
      + "        return False" + System.lineSeparator()
      + "    # NB: cx,cy are valid coords" + System.lineSeparator()
      + "    return cy == 0 or image[cy-1,cx] == 0 or cx == 0 or image[cy,cx-1] == 0 or cx
== w-1 or image[cy,cx+1] == 0 or cy == h-1 or image[cy+1,cx] == 0" + System.lineSeparator()
      + "" + System.lineSeparator()
```

Coding

Memory

- Use a lot of RAM: 2.5 GB, even for small images?
- Memory leakage
- Close the communication channel when changing the image?

Macro

- How to keep the encoded part?

Multiple channel?

-