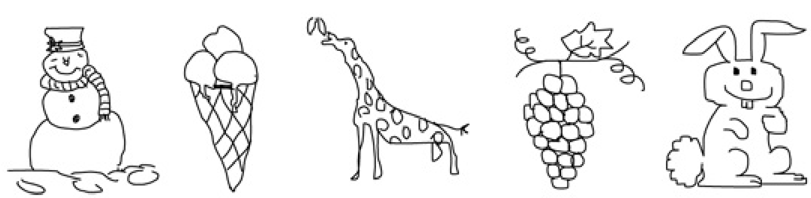
2951-B
Final Project

Ammar Hattab

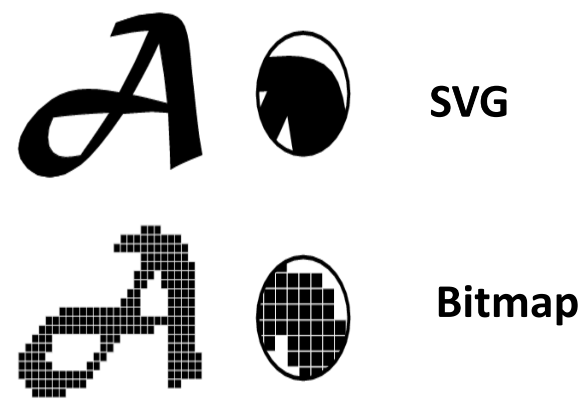
Sketch Recognition Using Vector Graphics

How Do Humans Sketch Objects?
Mathias Eitz, James Hays and Marc Alexa



+

Vector Graphics



Four Experiments

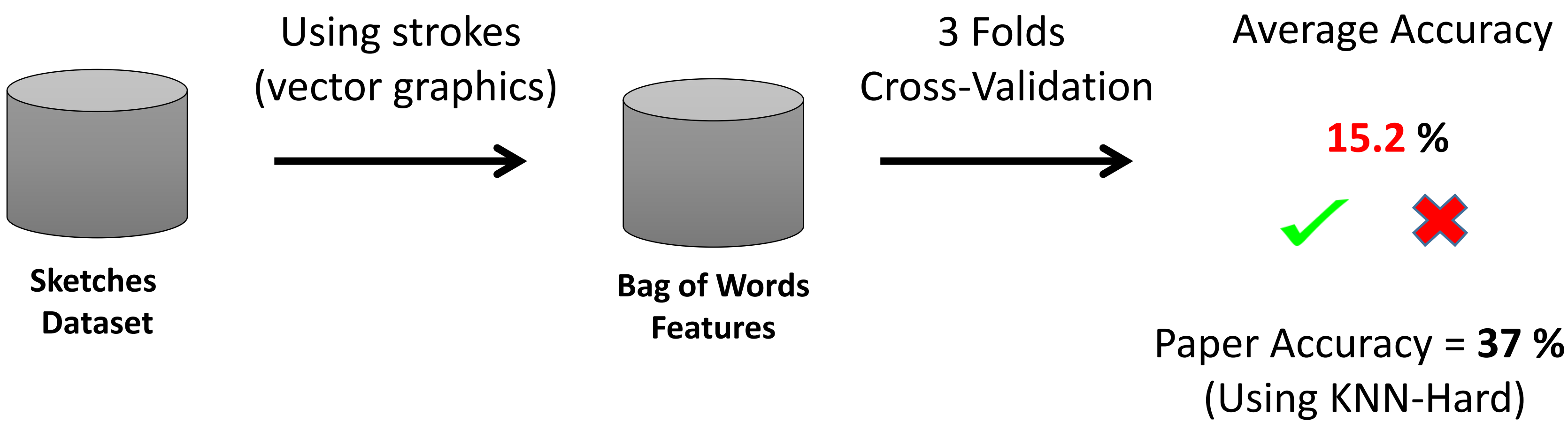
1. Paper's method, but using SVG instead of Bitmaps.
2. Adding Global Features
3. Curves Matching
4. Smaller Dataset

Experiment 1

Paper's Method: HOG + Bag of Words + SVM Classifier

But using **SVG**

Previously



- Something wrong !!

Patch Size

- Discovered the problem:
  - Patch_width = 12.5 % * sketch_width
  - Patch_height = 12.5 % * sketch_height
- Should be:
  - Patch_area = 12.5 % * sketch_area
- My patches were smaller !
- Fixing that increased the accuracy to:

15.2 % ✓ ✗ → 34.2 % ✓ ✗

Paper Accuracy = **44 %**
(Using KNN-Soft)

SVM Classifier

- Tried two implementations of SVM classifier:
  - LibSVM.Net
  - Accord.Net

34.2 % → **SVM Classifier** → **45.8 %**

- Using Paper’s Features:

52 %

Difference Reason

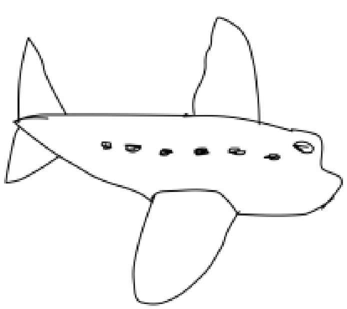
Bitmap Accuracy

52 %

SVG Accuracy

45.8 %

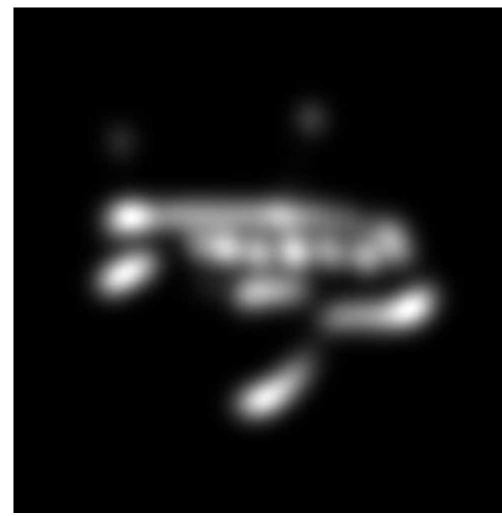
- By debugging paper's features:



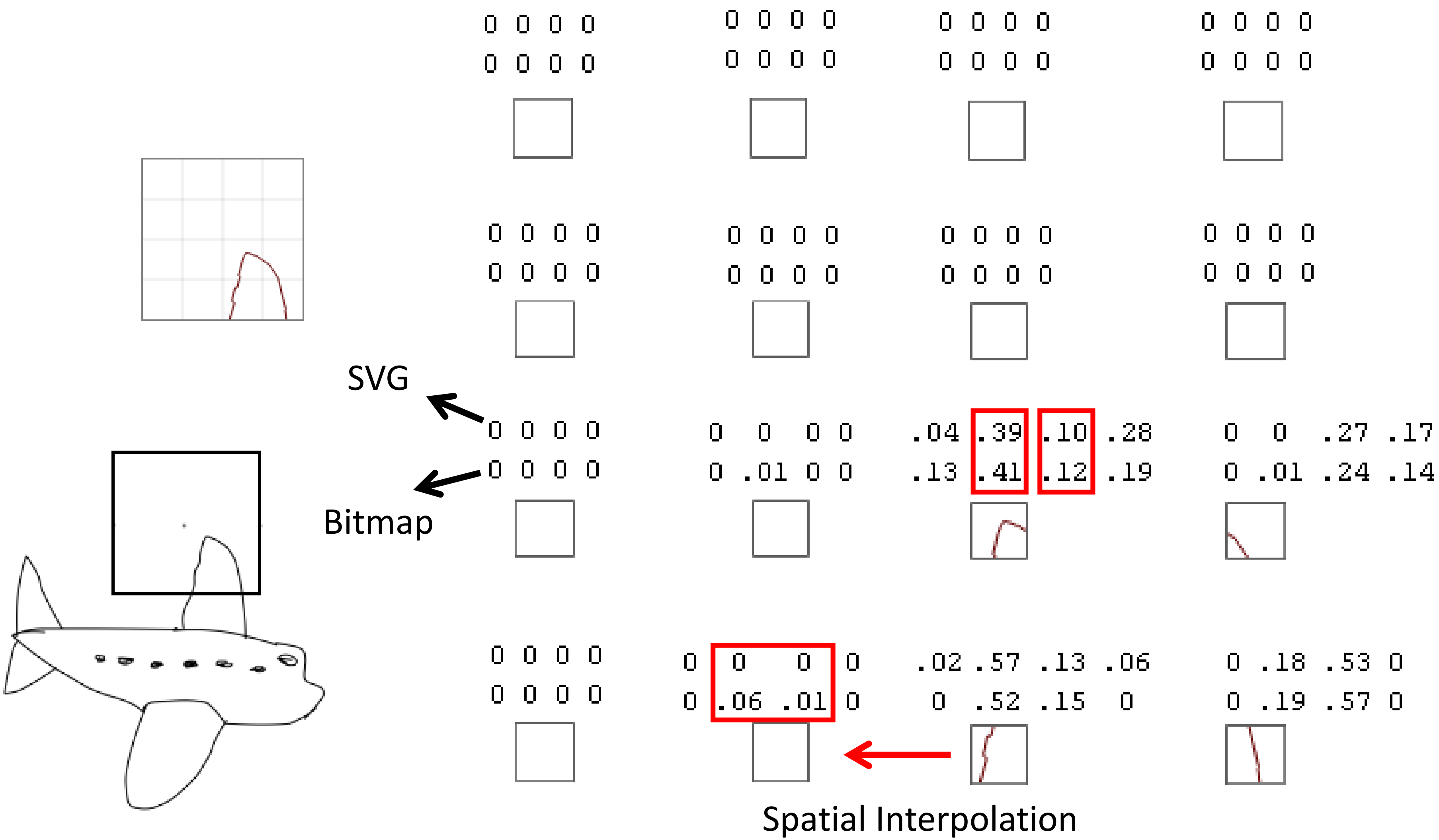
blur

gradient

Blur and increase size



Difference Reason



Difference Reason

- In conclusion:
  - We could use SVG to get the same **features** in the paper's method, using less computation.
  - We could get same **accuracy** (or worse) if we use SVG
  - but not better !
    - We don't need higher resolution for HOG,
    - in fact we need to **blur**!

Experiment 2

Adding Global Features

Global Features

- Features of the **whole shape**
  - (while local features are computed around a point)

- I used 3 types of global features:

  - Strokes Length
  - Points Counts
  - Moments Invariants

Global Features

- Image Moments:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

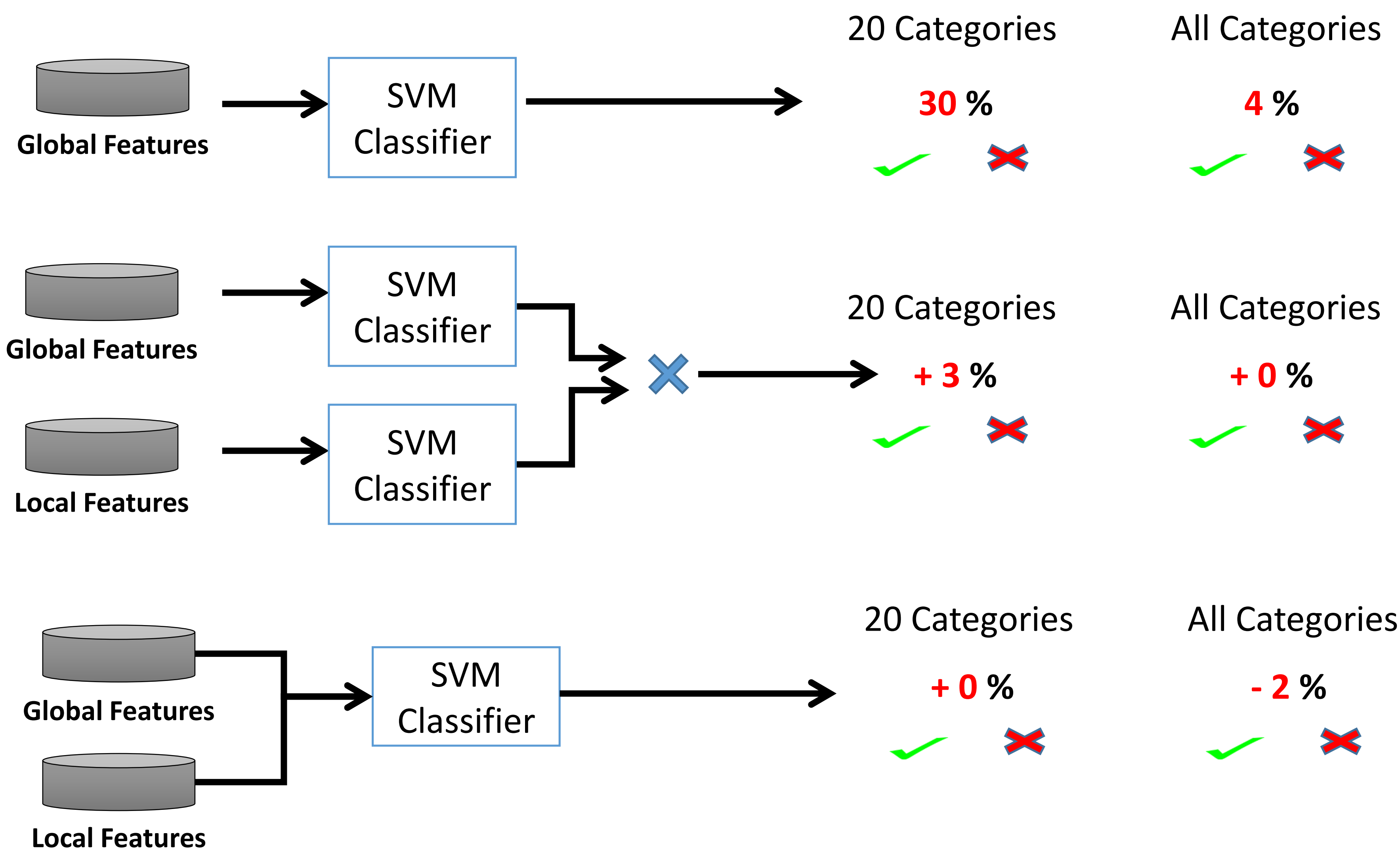
  - Describes the shape

$$M_{21} = \sum_x \sum_y x^2 y$$

  - Could be used to get: centroid, area, orientation, skewness, flatness…etc

- Moments Invariants:
  - Functions of image moments
  - Invariant to changes in (translation, scale, rotation)

Using Global Features



Conclusion

- Global features have small or no effect

- Possible Reasons:
  - Local features are strong enough
  - My choice of global features were weak.

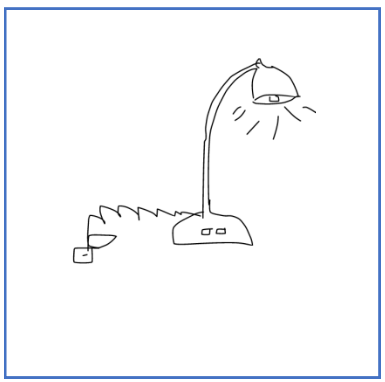
Experiment 3

Using Curve Matching

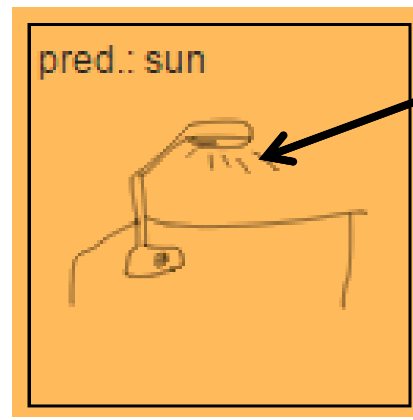
Curve Matching

- Local and global features do not care about the **spatial arrangement** or the **geometry** of the shape.

Table Lamp



Sun

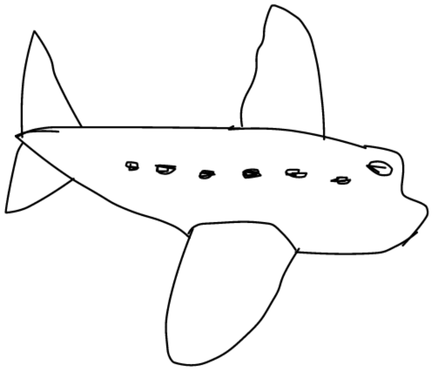


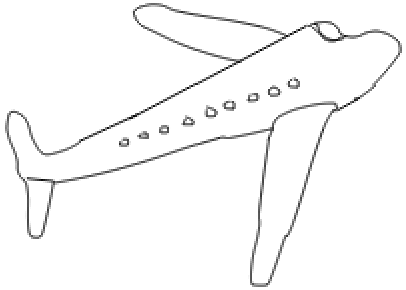
Same local Patches of sun

- Many wrongly classified sketches could be fixed by aligning and closely matching them to training sketches.

Iterative Closest Point (ICP)

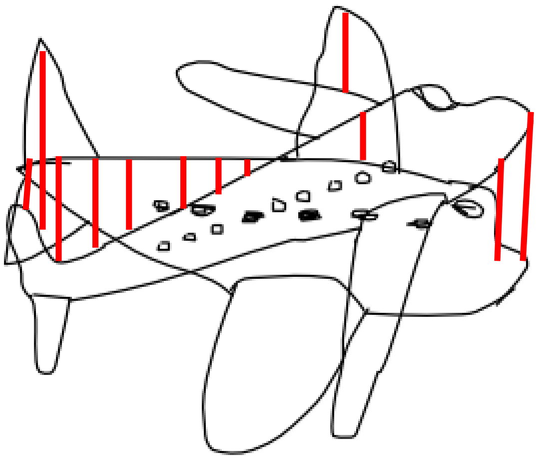
- To align two **point clouds** (set of points)





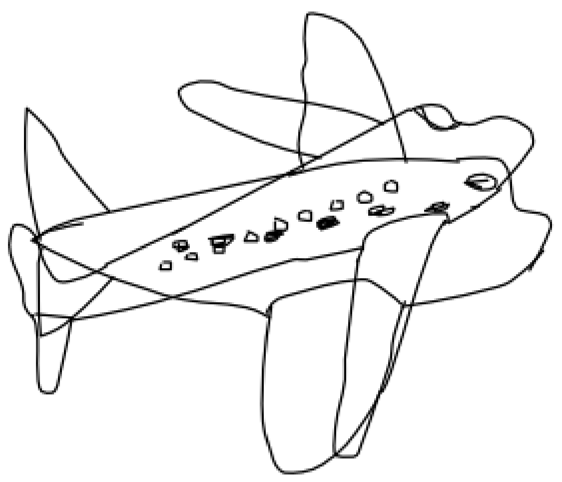
Iterative Closest Point (ICP)

- To align two **point clouds** (set of points)
- Iterate in two steps until finding the best alignment:
  - Find the closest points (to each point in the first)
  - Find the best alignment (and align them)



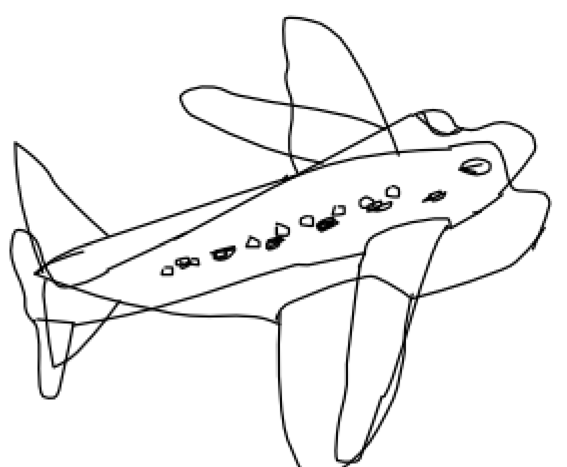
Iterative Closest Point (ICP)

- To align two **point clouds** (set of points)
- Iterate in two steps until finding the best alignment:
  - Find the closest points (to each point in the first)
  - Find the best alignment (and align them)



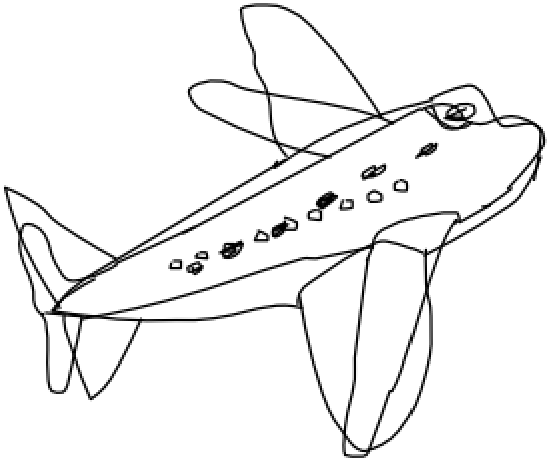
Iterative Closest Point (ICP)

- To align two **point clouds** (set of points)
- Iterate in two steps until finding the best alignment:
    - Find the closest points (to each point in the first)
    - Find the best alignment (and align them)



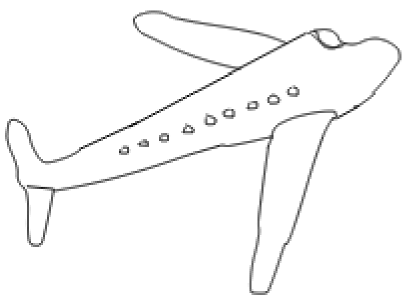
Iterative Closest Point (ICP)

- To align two **point clouds** (set of points)
- Iterate in two steps until finding the best alignment:
  - Find the closest points (to each point in the first)
  - Find the best alignment (and align them)

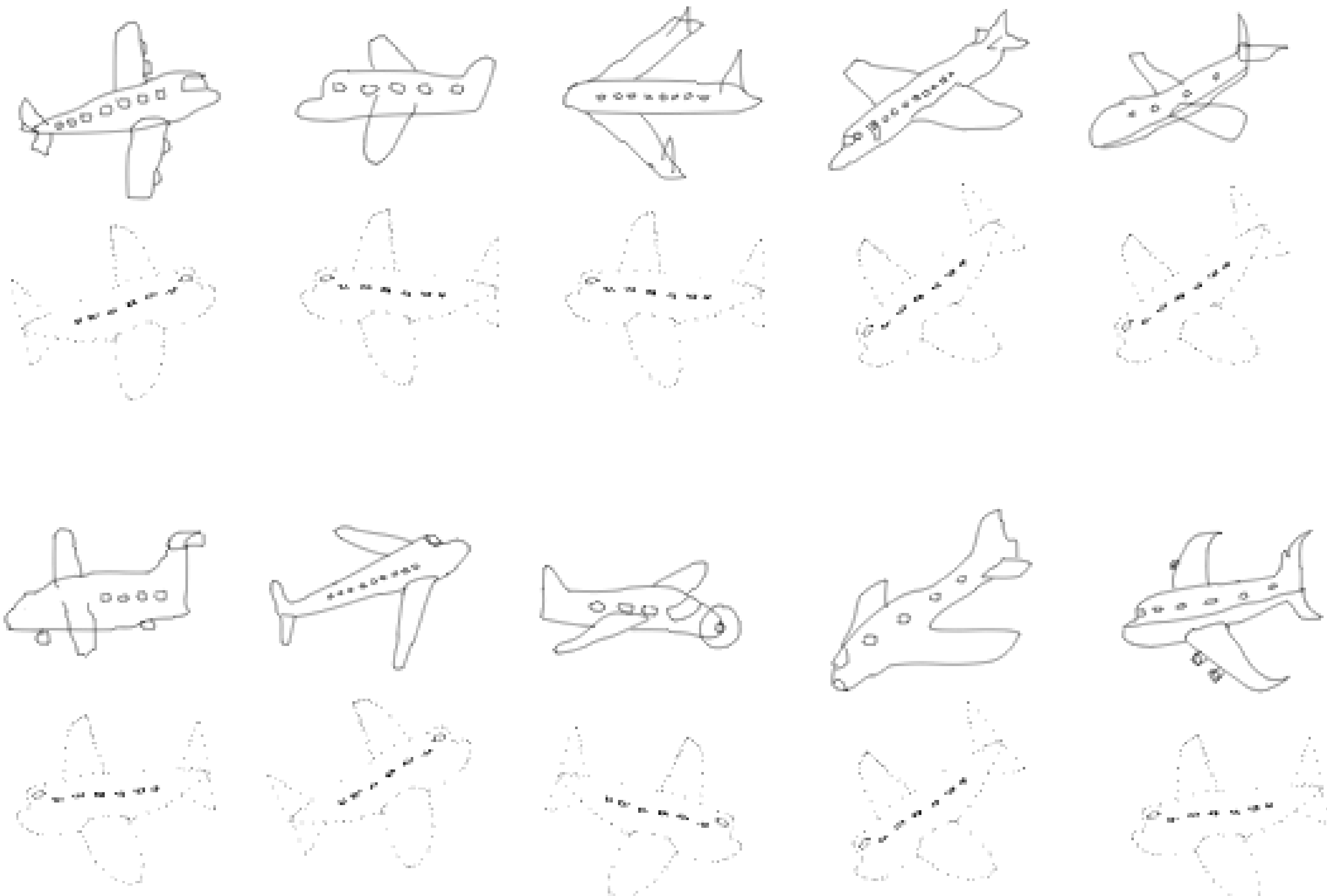


Iterative Closest Point (ICP)

- To align two **point clouds** (set of points)
- Iterate in two steps until finding the best alignment:
  - Find the closest points (to each point in the first)
  - Find the best alignment (and align them)



Examples



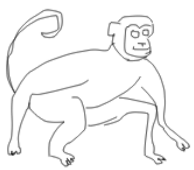
Algorithm

- Matching sketches one by one takes a long time
  - 20,000 X 20,000 = **277,7 days**

- When using local features:
  - correct category in the top 10 in : **80%** of the time

- We could only match the first 10 categories, instead of all categories

Algorithm Steps



Test Sketch

→ **SVM Classifier** →

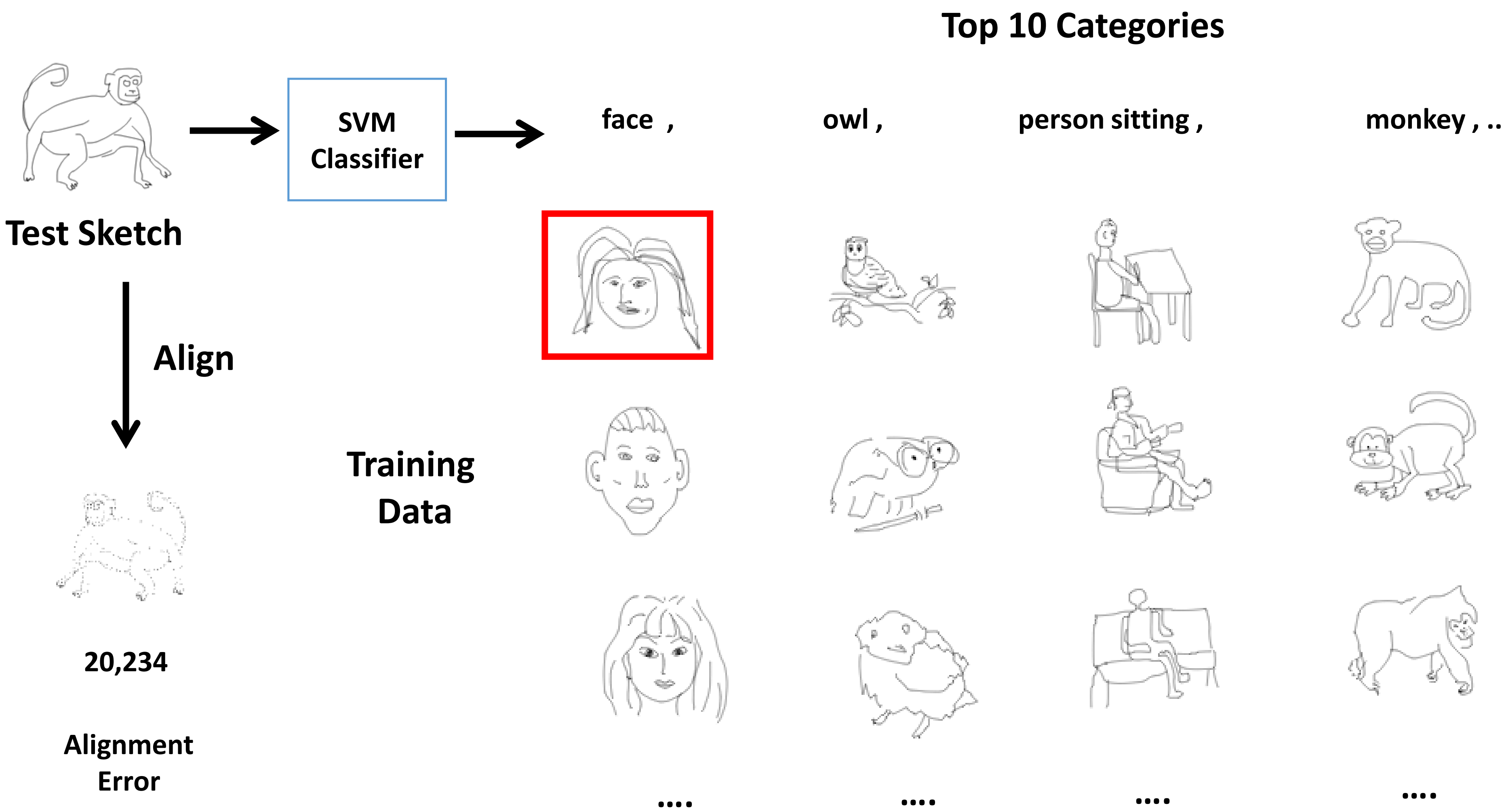
Top 10 Categories

face , owl , person sitting , monkey , ..

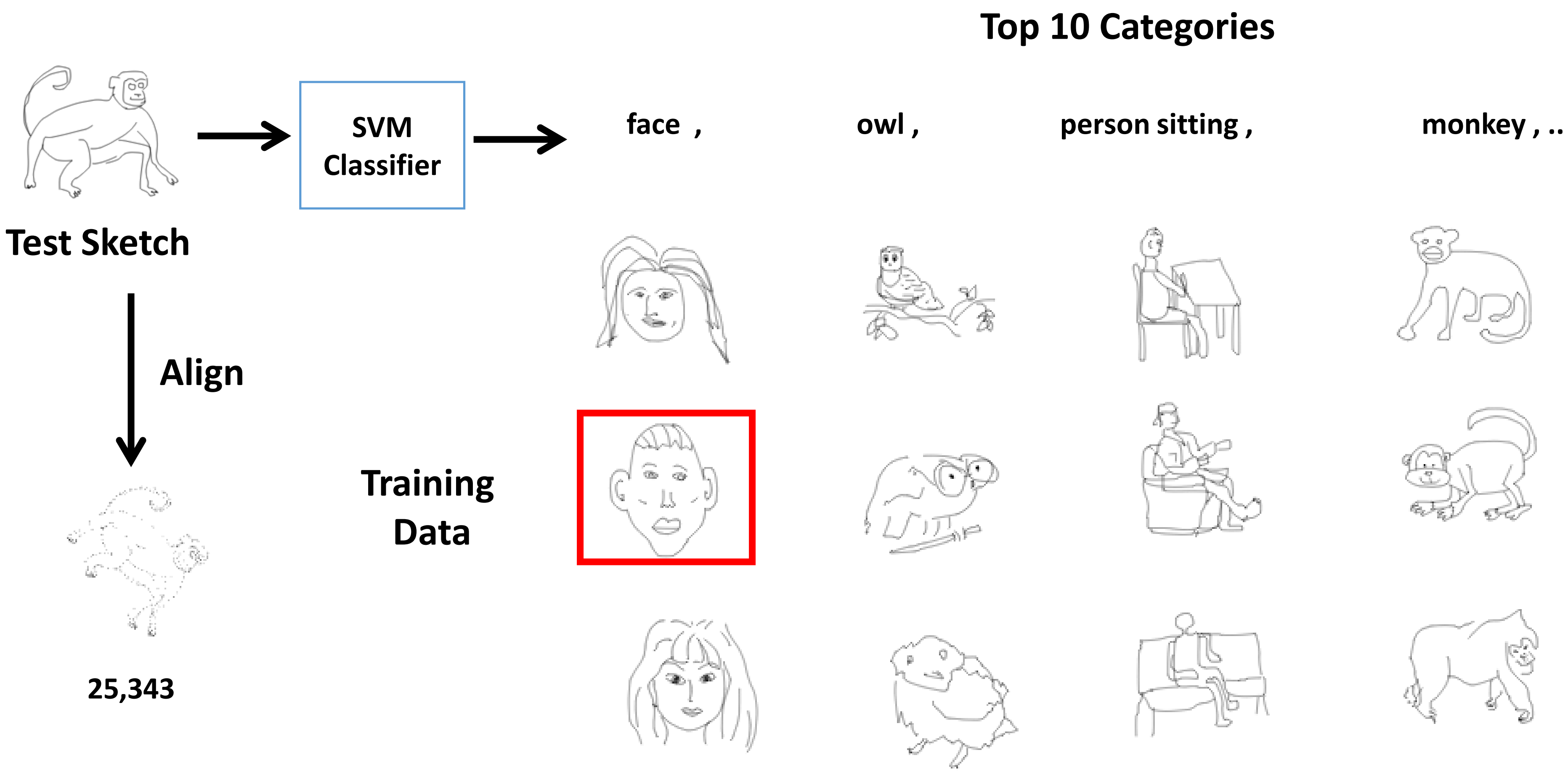
Training Data

....

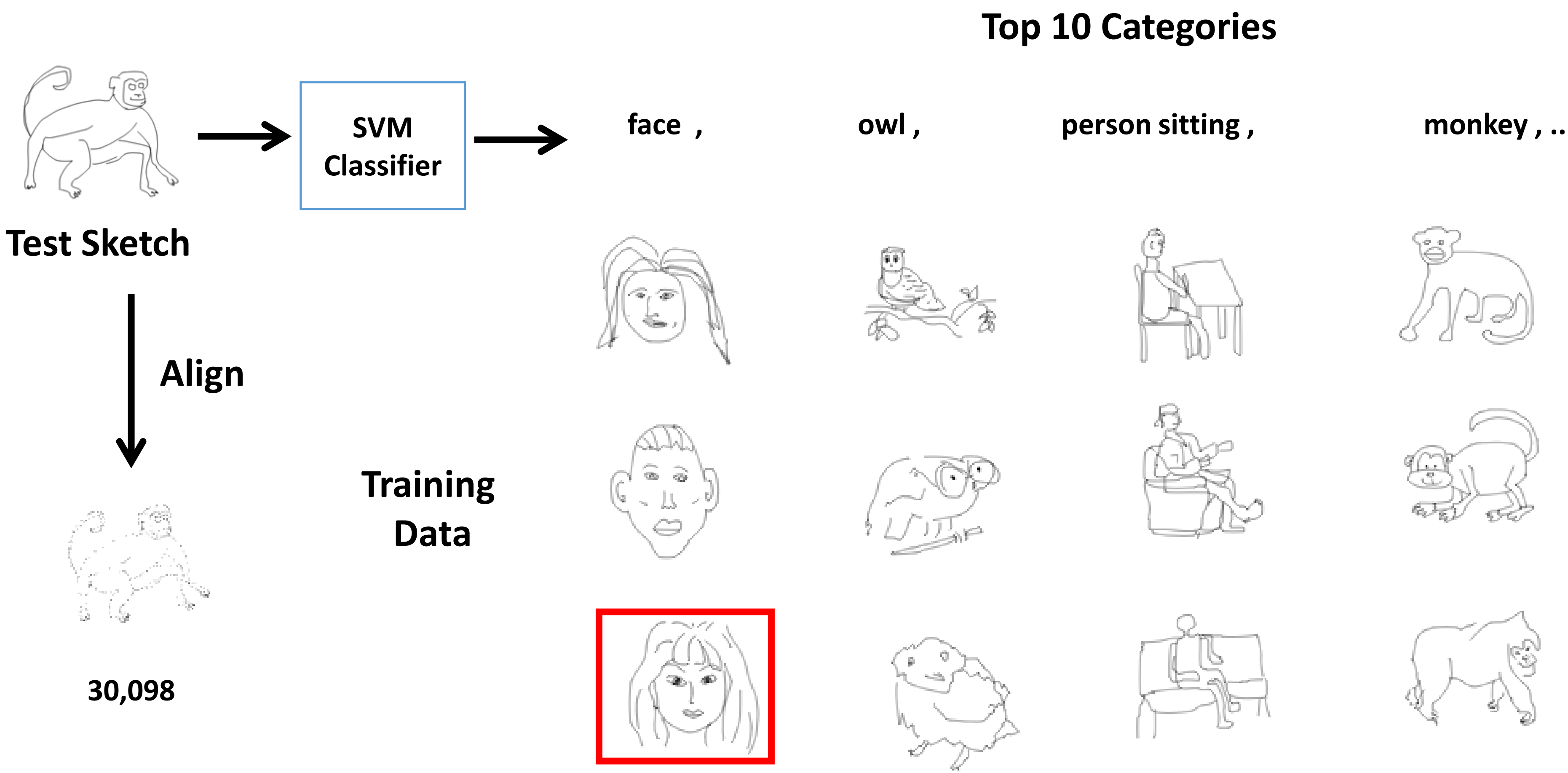
Algorithm Steps



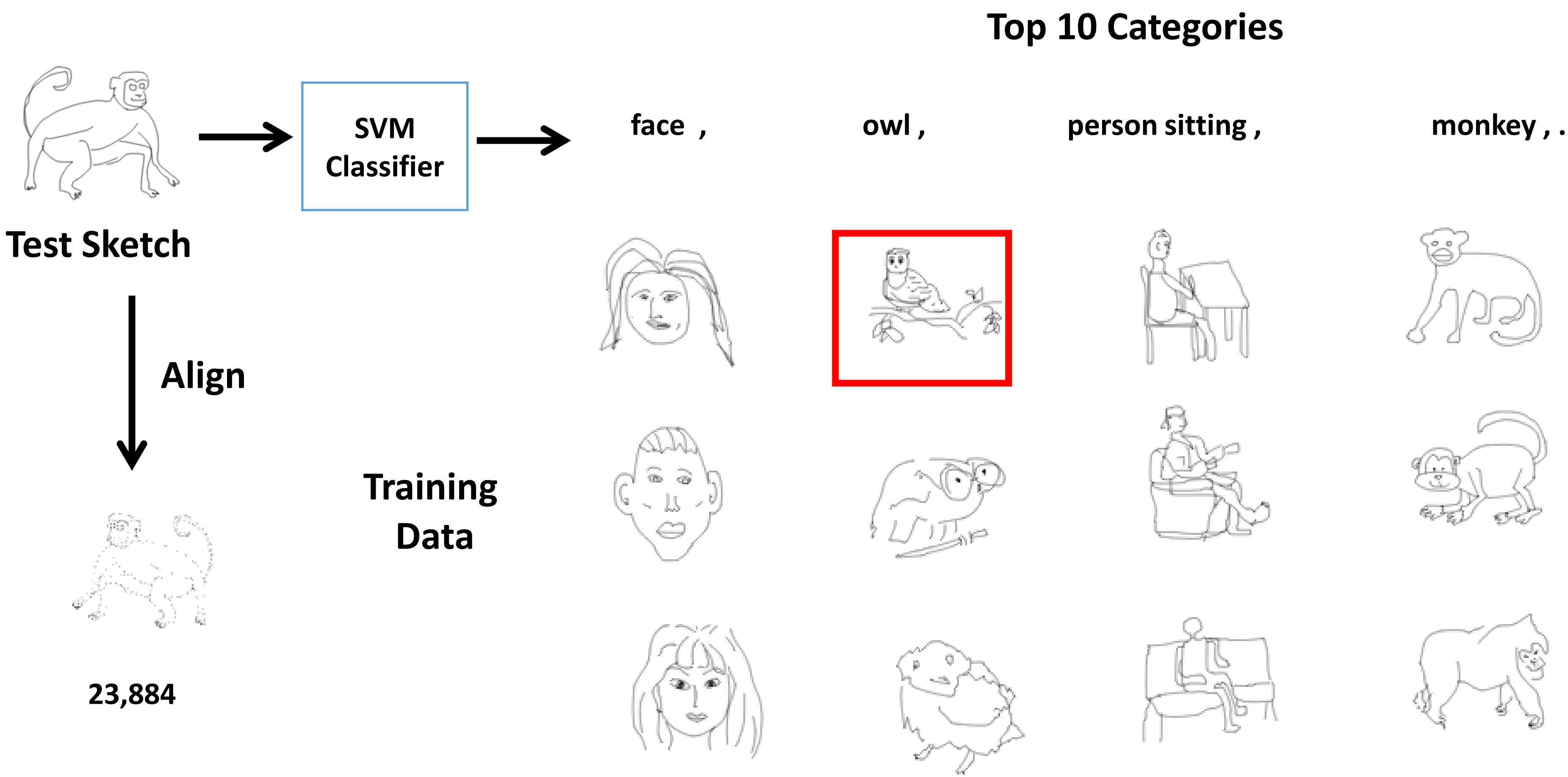
Algorithm Steps



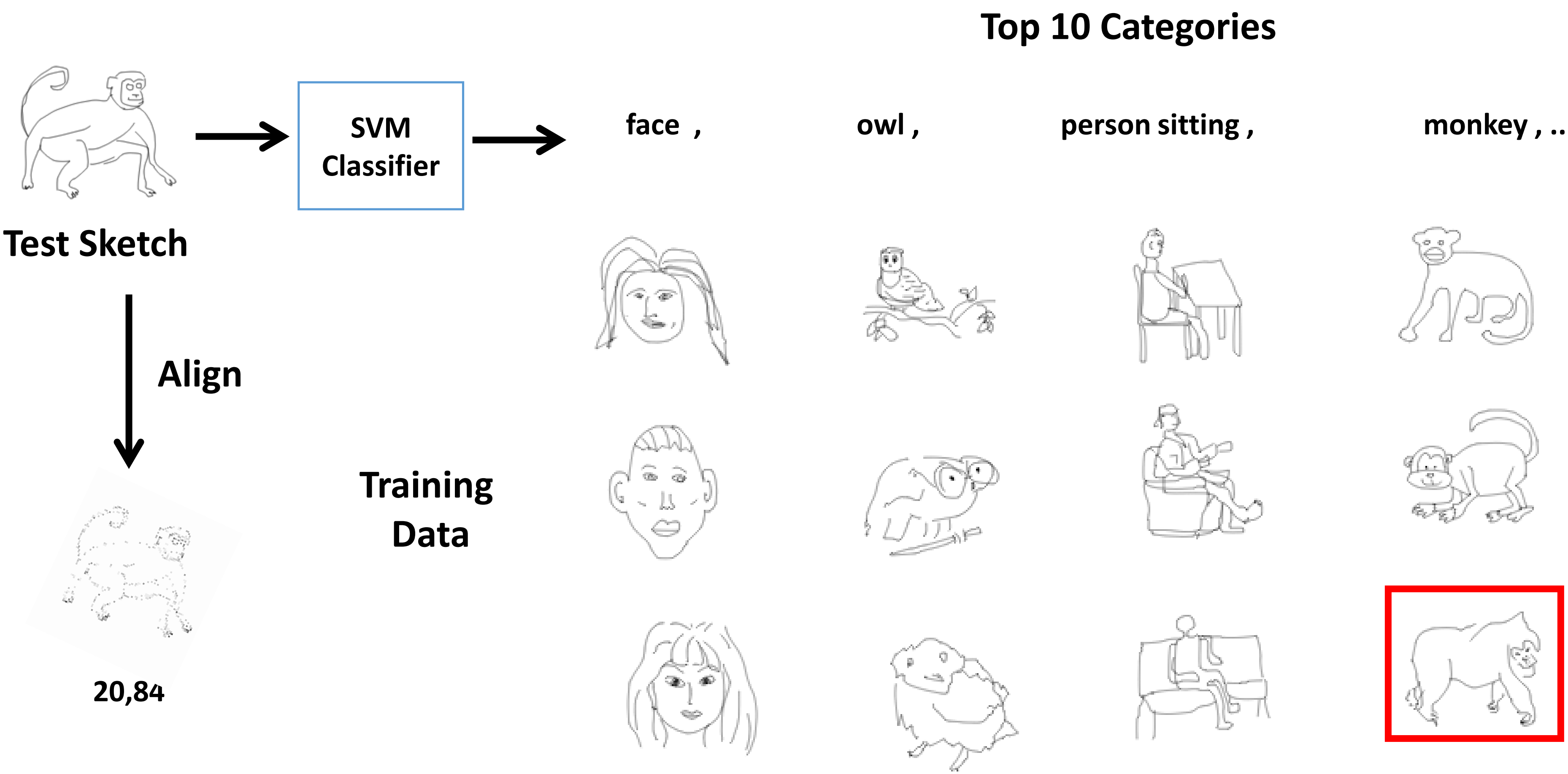
Algorithm Steps



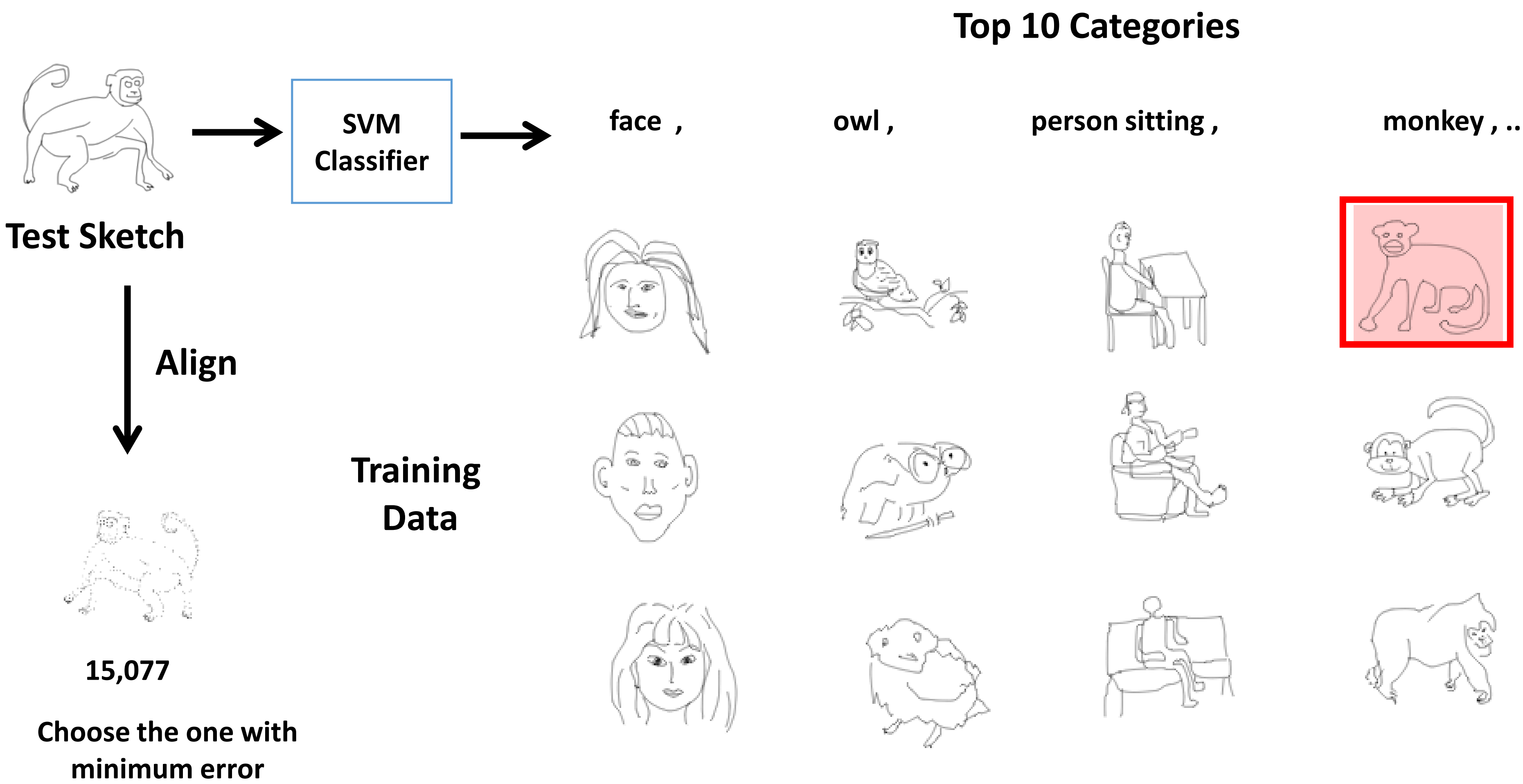
Algorithm Steps



Algorithm Steps



Algorithm Steps



Results

- Still it needs about 10 days to test all categories
- Applying it to the hardest category "**Monkey**"

7.4 % → 19 %

- Testing another hard category "**bottle opener**"

14.8 % → 26 %

Conclusion

- Closely matching the top categories will give **better accuracy** but **much longer time**

- For a **new sketch**, it takes about **1 minute** to classify it.

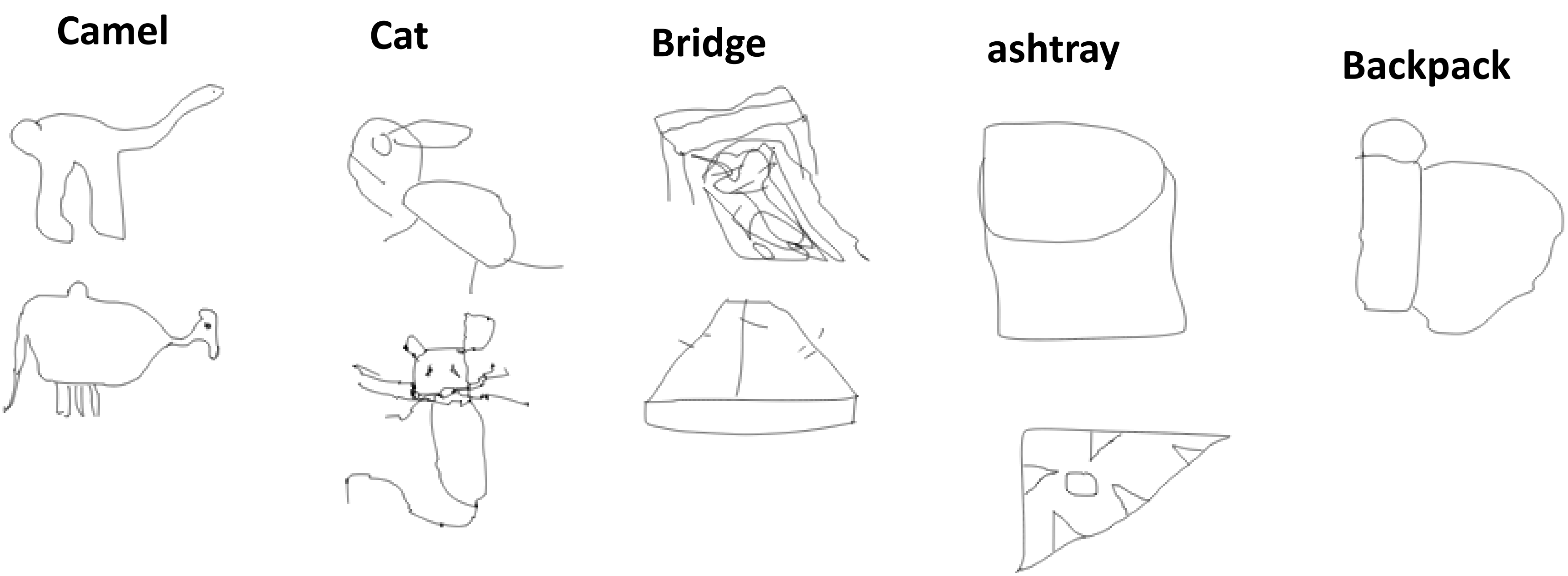
- But could be made faster by using **parallel** computing, or a faster and better **matching algorithm**.

Experiment 4

Smaller Dataset

The need for better data

- Many bad sketches cannot be classified
- The training database could be further cleaned



Smaller dataset

- I have manually selected the best 25 sketches from each category (~ 30%)
- Total of: 6250 sketches
- Cross-validation on the small sketches dataset:

52 %	**>**	**44 %**
Small-Dataset Accuracy		**Paper's Accuracy**

Thank you