STAT: Scene Text Annotation Tool

v 0.1
by Muhammet Baştan
mubastan@gmail.com

Date: 20 July 2015, Update: 01 Aug 2015, 12 Aug 2016, 15 Feb 2019

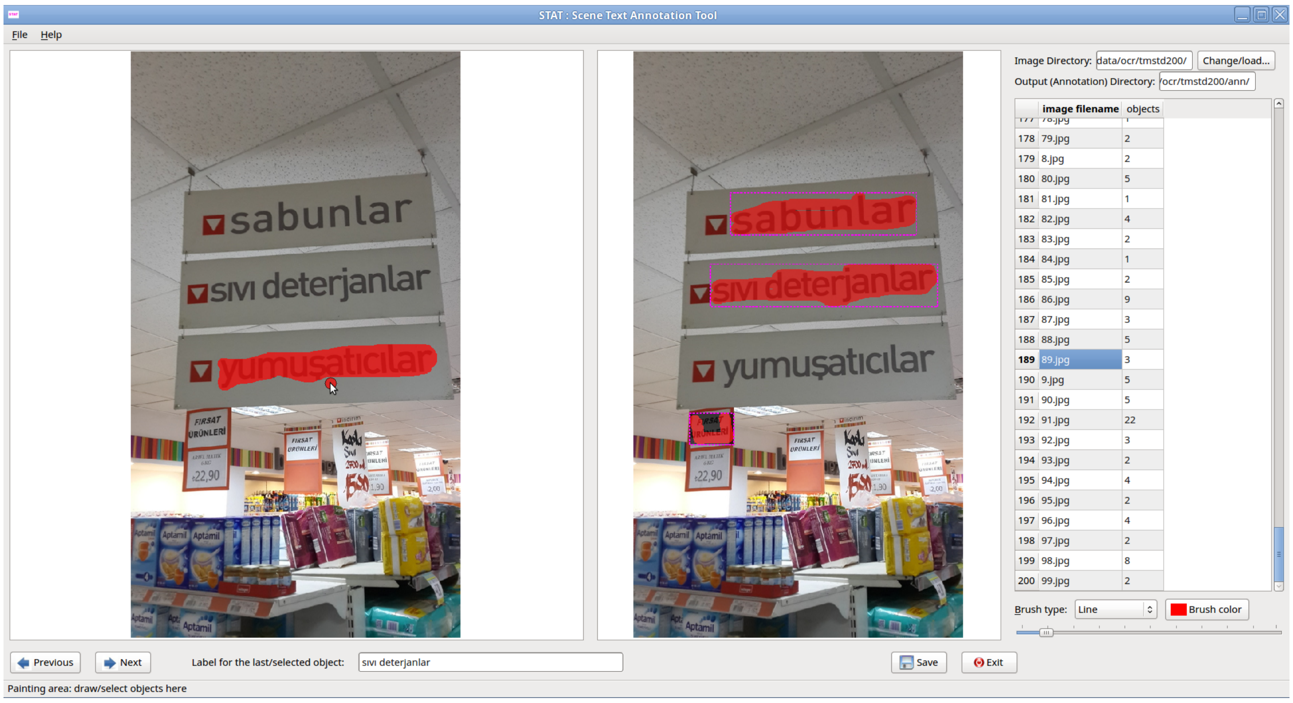

Image Directory: data/ocr/tmstd200/
Output (Annotation) Directory: /ocr/tmstd200/ann/

	image filename	objects
177	78.jpg	1
178	79.jpg	2
179	8.jpg	2
180	80.jpg	5
181	81.jpg	1
182	82.jpg	4
183	83.jpg	2
184	84.jpg	1
185	85.jpg	2
186	86.jpg	9
187	87.jpg	3
188	88.jpg	5
189	89.jpg	3
190	9.jpg	5
191	90.jpg	5
192	91.jpg	22
193	92.jpg	3
194	93.jpg	2
195	94.jpg	4
196	95.jpg	2
197	96.jpg	4
198	97.jpg	2
199	98.jpg	8
200	99.jpg	2

Brush type: Line

Label for the last/selected object: sıvı deterjanlar

Painting area: draw/select objects here

Figure 1: STAT screenshot

STAT is a simple **s**cene **t**ext **a**nnotation **t**ool (written in Python using PyQT4), to select and annotate **objects** or **text** in images. The object/text selection is either by painting over the object with the mouse, or by drawing a polygon. In both cases, the **minimum bounding box** and the **object mask as a bitmap** are saved on the disk. The objects can be labeled; the labels can be UTF-8 text (tested for Turkish, but not for any other language).

Version 0.1 of the tool is developed specifically to annotate scene text regions, but it can also be used to annotate other object categories.

The annotations are saved as text files (+ .png files for object masks). It can be easily modified to save in xml format, if needed.

1. Installation

Required: Python 2.7, PyQT4
Developed and tested on Linux Mint 17.1.
Open a terminal in the directory of the tool and just type: `python STAT.py`
Then, you should see the above GUI (without any images loaded).
It is adapted from the X-ray Image Annotation Tool (XRanT), and the current

code is unfortunately not in a good state; feel free to contribute to improve it.
XRanT: https://sites.google.com/site/mubastan/research/software/xrant

2. Using STAT

The images to be annotated need to be in a single directory. Then, open this directory to load the list of images in one of these ways:
>> File > Open Image directory
>> Ctrl + O
>> Click on Change/load button at the top right corner

The annotations will be saved in a subdirectory called "ann" under the image directory. So, make sure you have write permissions for the image directory otherwise the annotations will not be saved.
Planned: Changing the output directory (You can open Ann.py with a text editor, go to line 'self.annotationDir = str(imageDir + '/ann/')' and change the directory to anything, e.g., self.annotationDir = '/home/ali/annotations/')

The GUI has 3 main panels:

- Image panel/canvas on the left: here you select the objects with the mouse (painting, polygon). Right click to see what operations are available (add object, hide/show brush,...)
- Image panel in the middle: this shows the currently selected objects. You can right click on the objects to update their label or delete them. You can right click on the background to delete all objects, etc.
- The table on the right: shows the list of all images in the image directory, and number of objects selected.

Hint: hover you mouse over the GUI elements to see short help texts on the status bar.

Button **next** (shortcut: **space**): go to the next image; current image annotation (bounding boxes + bitmaps) is saved, if anything has changed since last save.

Button **previous** (shortcut: **Ctrl+space**): go to the previous image; current image annotation (bounding boxes + bitmaps) is saved, if anything has changed since last save.

Button **save** (shortcut: **Ctrl+S**): save current image annotation (bounding boxes + bitmaps).

Text field at the bottom: shows the label of the currently selected object. You can change the label of the currently selected object by entering the new label and pressing Enter (or right click on the object and select 'update label').

Brush type combo box: for selecting the brush type. You can also change the brush color by clicking 'Brush color' button.

Slider: for selecting the width of brush for painting.

Button **Exit**: Exits the application, by first confirming, then saving if required.

Since everything is saved on the disk, when you re-load the image directory after you close and re-open the application, it checks the already available annotations in the annotation directory and loads them automatically. So, you can do annotation in multiple sessions, you can update annotations, etc.

3. Outputs

Outputs are saved to the annotation directory.

1. Object masks, as png files; one image for each object: image.0.png, image.1.png,...

2. Label files: bounding box and label for each bounding box, to a txt file for each image: image.png.labels.txt

Example: image.png.labels.txt

```
image.png
0 6 246 65 34 Text1                    # objectID leftCoord topCoord  W H Label
2 81 75 89 35 Text2
4 15 70 67 29 Text3
1 8 181 65 26 Text4
3 14 122 103 30  Text5
```

3. Only the bounding boxes into a txt file for each image

Note: this is done only when you do: File > Save all annotation bboxes

Then, this file is saved for all images.

Example: image.png.bbox.txt

```
/home/research/code/imant-py/text/image.png
238 293             # ImageWidth ImageHeight
6 246 65 34         # leftCoord topCoord  width height
81 75 89 35
15 70 67 29
8 181 65 26
14 122 103 30
```