Text2Scene: Generating Compositional Scenes from Textual Descriptions

Albert Nasybullin (a.nasibullin@innopolis.university), Marina Lisnichenko (m.lisnichenko@innopolis.university)

Project idea: to rebuild text into image

Dataset: Abstract Scene Dataset v1.1

Timeline with each member individual task:

- 1st week: literature review and code investigation
- 2nd week: object detection
- 3rd week: code creating
- 4th week: cartoonizer
- 5th week: presentation

References:

Original paper, their previous paper

What do we want:

As far as some code was found in the GitHub, firstly we want to test it and find out either it is workable or not. If there is no problem, we can suggest to do the following work: on synthesized image find objects, image of which separately are saved in the dataset (just pattern recognition).

Then there is an idea to move some objects. For example, when sentence is “Mike throws a ball to Jenny” ball is moving from Mike to Jenny.

Final step is to generate from the output image a comic-book style picture. We propose to creater another GAN for this task.

In the end it could be done text-to-image work. For example, when ball appears in the Jenny’s hands the output sentence is “Now Jenny holds ball”.

But to do all of this work we firstly have to be sure, that primary scripts from Git are workable.

Final Update (26.04.2020)

What do we have till now:

- ☑ literature review
- ☑ Text2Scene algorithm (NOT from paper)
- ☑ Comic-book style picture

Problems

Unfortunately, we couldn’t run git from source paper. It happened because of (1) too slow internet to download around 100 Gb of info and (2) there is no instruction about how to run the code.

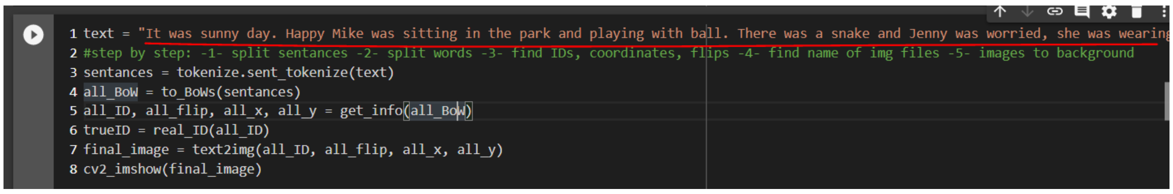
Text-to-scene algorithm was done from scratch by Marina. There is no Neural Networks, but, was done attempt to save the idea of authors of paper (make a code from text -> from code take image -> place images in the background). Because of this unexpected task, following moving objects wasn't done yet.

However, by Albert was done GAN, making comic-book style of pictures. Till now, it needs some debugging process, but already we have some good results.

Results

The code is there:
https://colab.research.google.com/drive/10a2mwlpcp9hURDLwGuWdUHrRVY88vnJO#scrollTo=Z3B6G3xFBbRm

Also you can check our git: https://github.com/UralmashFox/CV_Project
there is "CV_project.ipynb" file. All you need is to open it (for example in collab), in the last cell input the desired text:

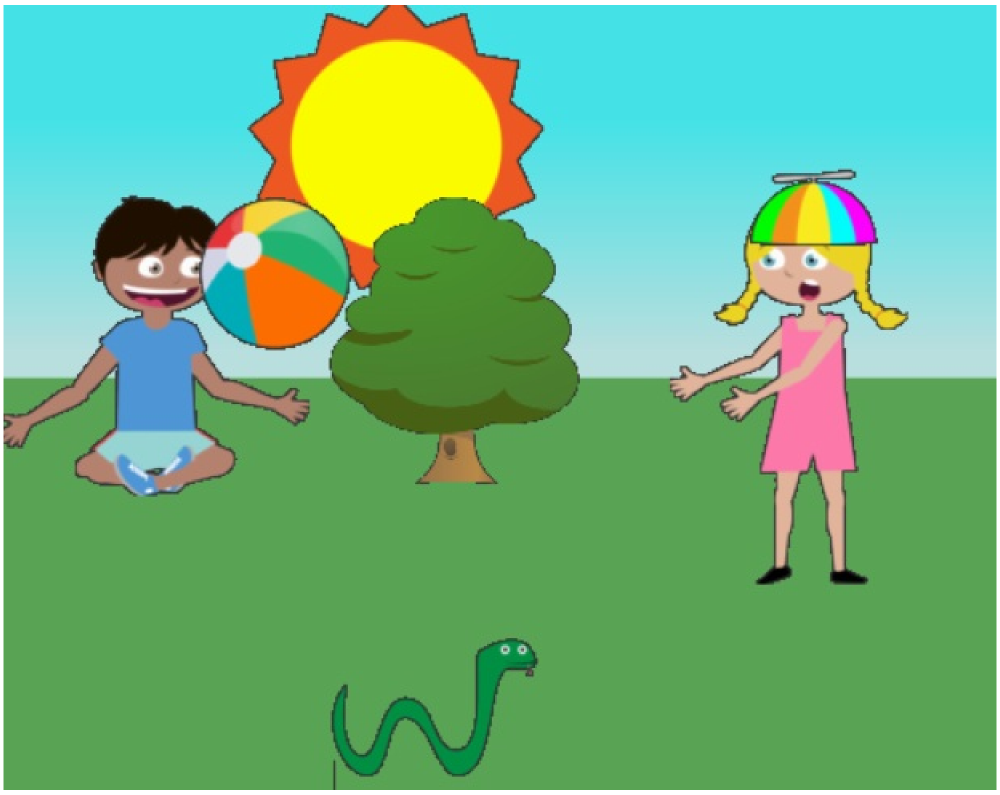


```
text = "It was sunny day. Happy Mike was sitting in the park and playing with ball. There was a snake and Jenny was worried, she was wearing
#step by step: -1- split sentances -2- split words -3- find IDs, coordinates, flips -4- find name of img files -5- images to background
sentances = tokenize.sent_tokenize(text)
all_BoW = to_BoWs(sentances)
all_ID, all_flip, all_x, all_y = get_info(all_BoW)
trueID = real_ID(all_ID)
final_image = text2img(all_ID, all_flip, all_x, all_y)
cv2_imshow(final_image)
```

For example, input

It was sunny day. Happy Mike was sitting in the park and playing with ball. There was a snake and Jenny was worried, she was wearing silly hat

will give you a following picture:



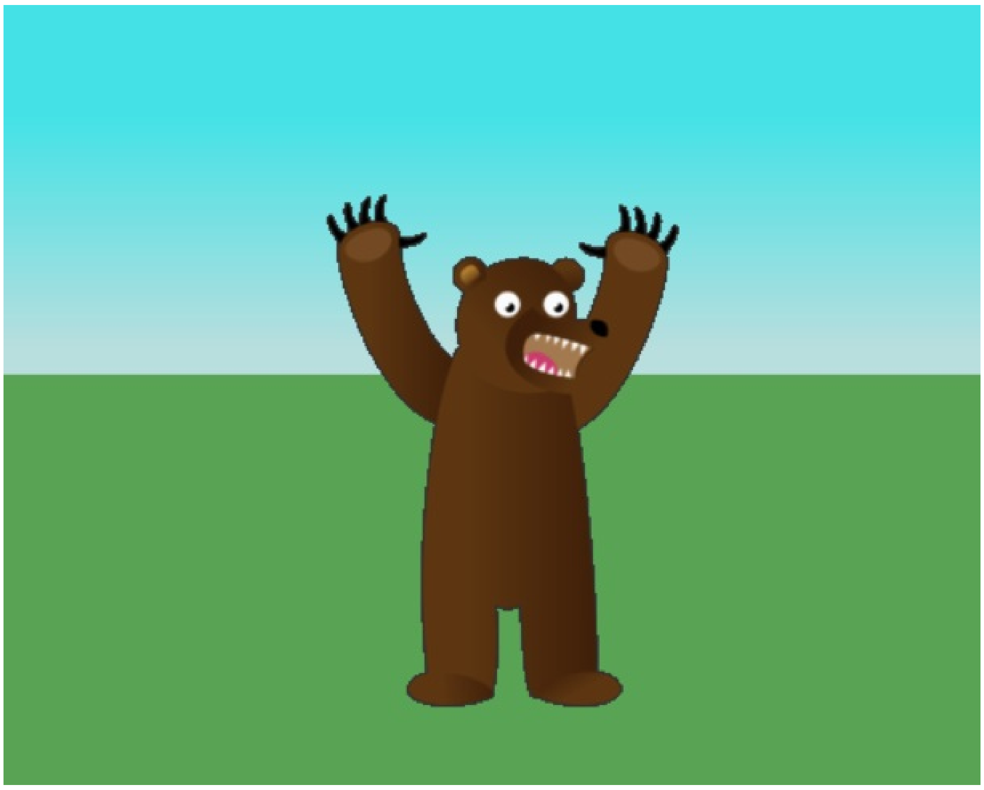
if you are going to change text, from second time run ONLY the last cell, not all the program.

Important issues

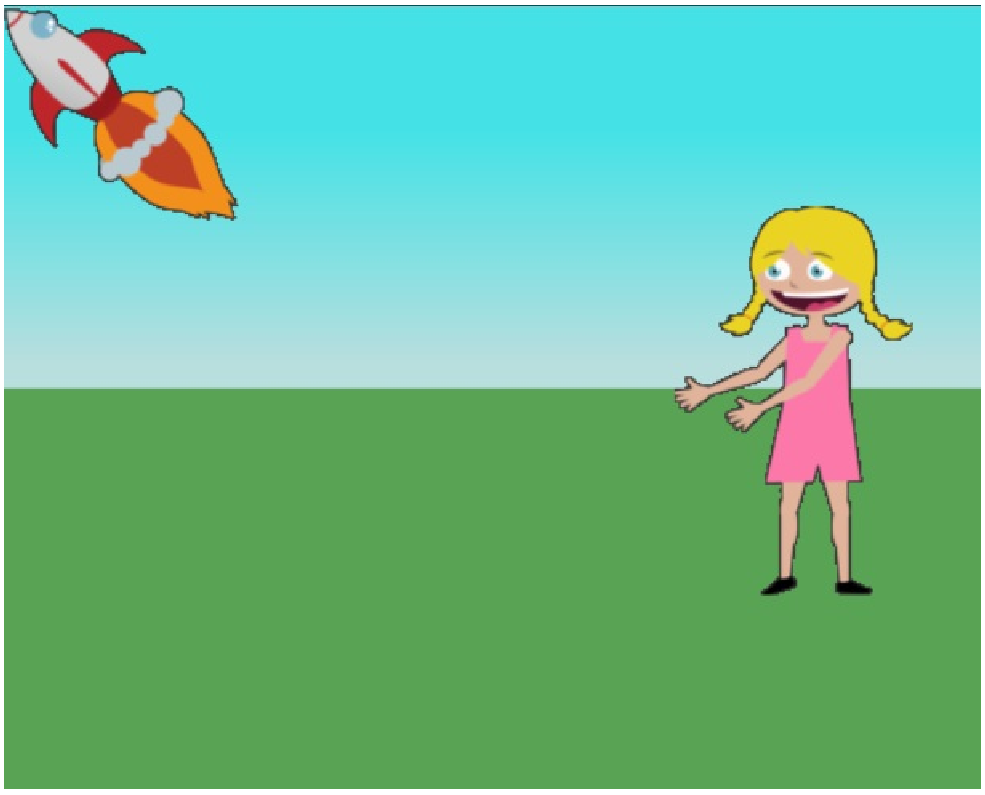
1. try to build picture in following steps for each sentence:
   sky objects -> ground objects -> person + mood + hat + activity -> animals.
   In case to not overlap small objects by big
2. follow syntax rules of English
3. you can input any amount of sentences with any amount of words
4. NOT input sentence with only hat or activity

Some examples

There is a bear

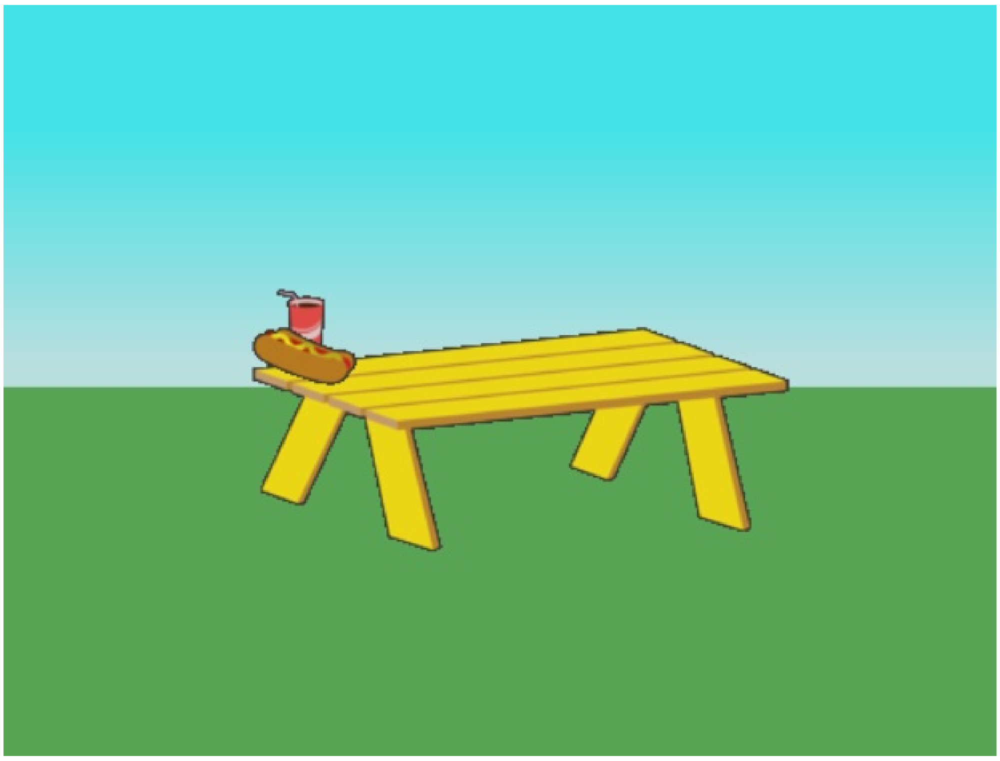


The rocket was flying in the sky and Jenny was excited

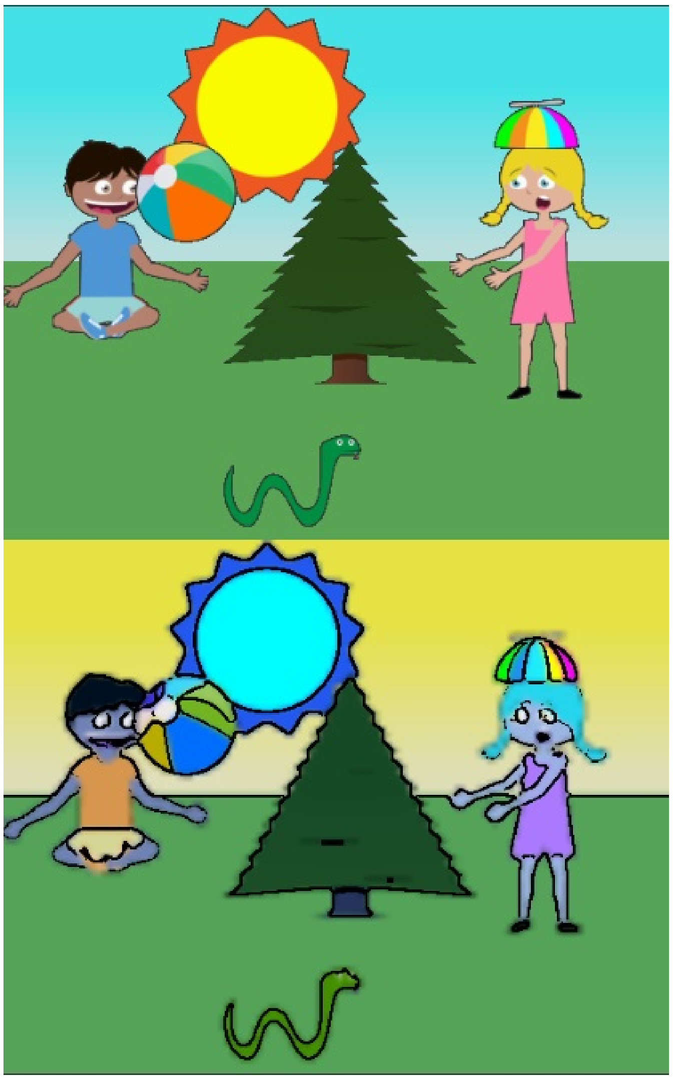


It was a storm. Despite of this, kids were in the park and playing soccer. Mike was sitting near sand. Jenny was a happy princes, she was kicking soccer

On the table there is a drink and hotdog



Also when you run the code, in the end there is a cartoonized image. For example:



Part2: Cartoonizing filter:

In the beginning, we wanted to create GAN, which will generate comics-like images from the images we get from the text2image part. Regretfully, we failed on this part. Mostly because of computation complexity and my personal (Albert's) lack of expertise in Generative Adversarial Network. Also, we realized a GAN approach is overcomplicated for our goals. So, we decided to create a filter based on classical Computer Vision methods.

We could've to use a pre-trained model for cartoonizing, but it doesn't sound like fun. Isn't it? So, we have spent most of our time trying to make the GAN works but failed. I will be happy to try it one more time during the summer holidays.

How cartoonizing filter works?

Papers and articles we found useful in this work:

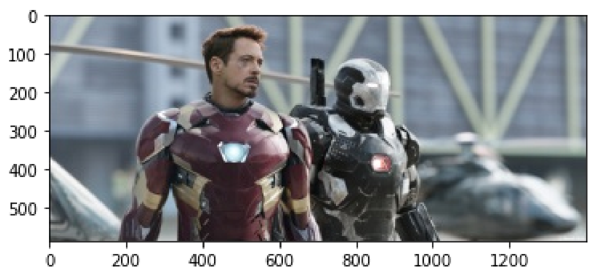
- Learning the K in K-Means,
- Converting Color Real Image to Cartoon Image Using NonParametric Mean-Shift Technique,
- Introduction to Image Segmentation with K-Means clustering,
- Color Quantization with OpenCV using K-Means Clustering,
- Learn How To Do K-Means Clustering On An Image,
- OpenCV with Python By Example;

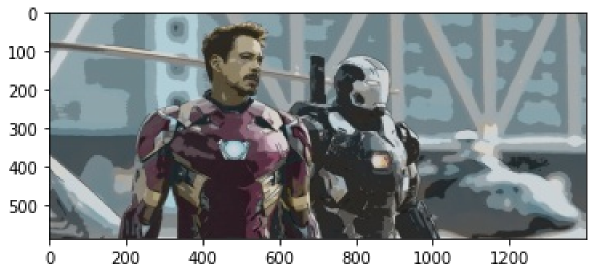
Step by step:

- to create histogram for given image,
- to find centroid,
- to update centroids as long as they won't stop changing,
- to choose minimum group size, alpha value to get the best value of k,
- to calculate values of H, S and V,
- to extract contours,
- to fill contours with calculated H, S, V;

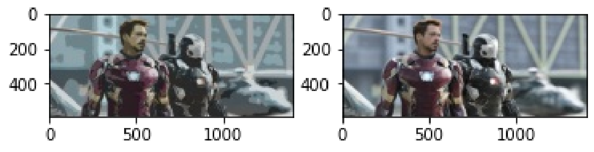
Given results of cartoonizng filter:

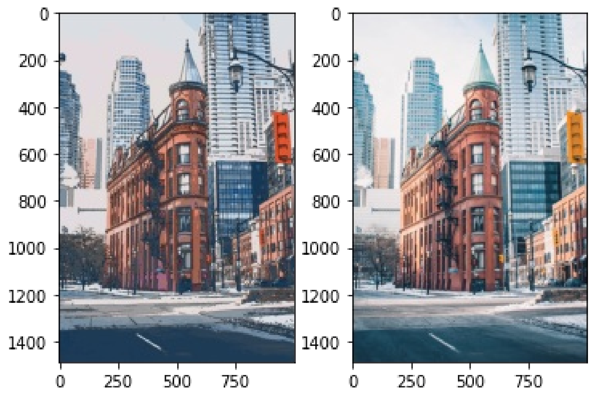
Regretfully, we didn't get enough attention to a fact, that Abstract Scene Dataset is very abstract itself. So, there is no real sense in cartoonizing abstract images. But our filter works pretty good for another type of image. For our image this filter gives us a way more shape contours. You can see several examples here:

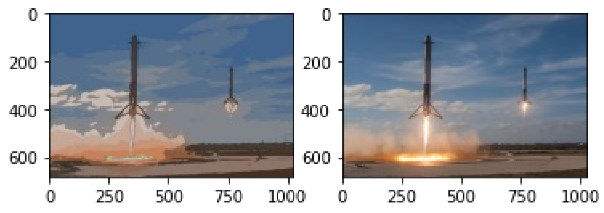




Another examples:







Due to computation limits cartoonizer part represented as temporary approach required for demo. It is supposed to be developed in the next time

We can see that after cartoonizing image looks more painted by hand, the result that we wanted is achieved.

Conclusion

Due to doing this project we met some problems in case to modify the goal and some tasks. In the end we have workable code from scratch that realizes most tasks which we planed.

If you're going to modify the text, PLEASE run*run all* only ONCE! When modifying text - change it and run only LAST cell! Thanks.