512-AS3-REPROT

Chen Xu

A20377739

1. Problem statement

1. Load and display two images contains similar content or capture by webcam.
2. Change all the images into grayscale before processing
3. Include the help key describing its functionality.
4. Estimate image gradients and apply Harris corner detection algorithm.
5. Obtain a better localization of each corner.
6. Compute a feature vector for each corner were detected.
7. Using the feature vectors, you computed match the feature points. Number corresponding points with identical numbers in the two images.
8. Interactively controlled parameters should include in functions.

2. Proposed solution

1. Harris corner detection:

$$R = \det(M) - k(\mathrm{trace}(M))^2$$

where

- $\det(M) = \lambda_1 \lambda_2$
- $\text{trace}(M) = \lambda_1 + \lambda_2$
- λ_1 and λ_2 are the eigen values of M

If R > threshold , we mark it a corner.

2. Compute a feature vector for each corner were detected:

    a. find the keypoints and descriptors with SIFT

    SIFT keypoints of objects are first extracted from a set of reference images[2] and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches.

    b. Use BFmatcher to compare the point you detected and SIFT descriptor finded. And number corresponding points with identical numbers in the two images.

    Brute-Force matcher is simple. It takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. And the closest one is returned.

3. Better Localization:

   Project gradients onto edge hypothesis and choose p with minimal projection.

   The location of corner:

$$p = c^{-1}v = c^{-1}\sum \nabla I(p_i)\nabla I(p_i)^t p_i$$

$$c = \sum \nabla I(p_i)\nabla I(p_i)^t$$

   c is nonsingular because $\lambda \cdot \lambda_1 > \tau$

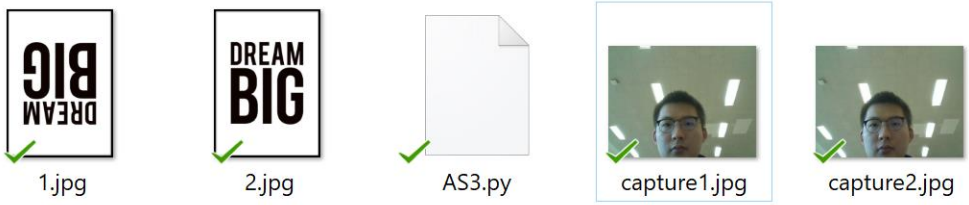
3. Implement details

1. Use webcam take two photos also give you two similar pictures.
2. Use an isolate display function to display result for every function.
3. When take parameter into some function, we need to know the type and change it into right type to compute.
4. If use trackbar to implement the real-time parameter changing will require the complete rebuild of functions. Therefore, I choose input parameter not trackbar.
5. Parameters are depending on the pictures you choose; every picture should pick parameter by judge the texture or complexity of picture.

6. It's possible to set different threshold at real-time, by store the 'r' and judge it every time you change the threshold. I left it in the code, but use the input way, because I want all the parameter can be changed at same time.
7. Press 'H' to get help of this program.
8. It required input parameter to get the result of Harris function.

4. Results and discussion

1. Load two picture or capture by webcam



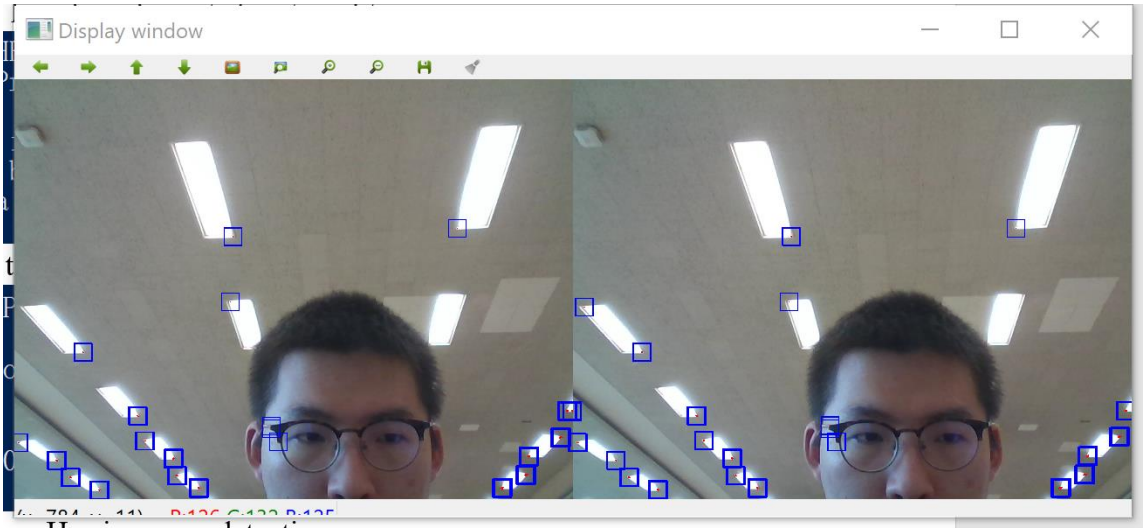
2. Include the help key describing its functionality.

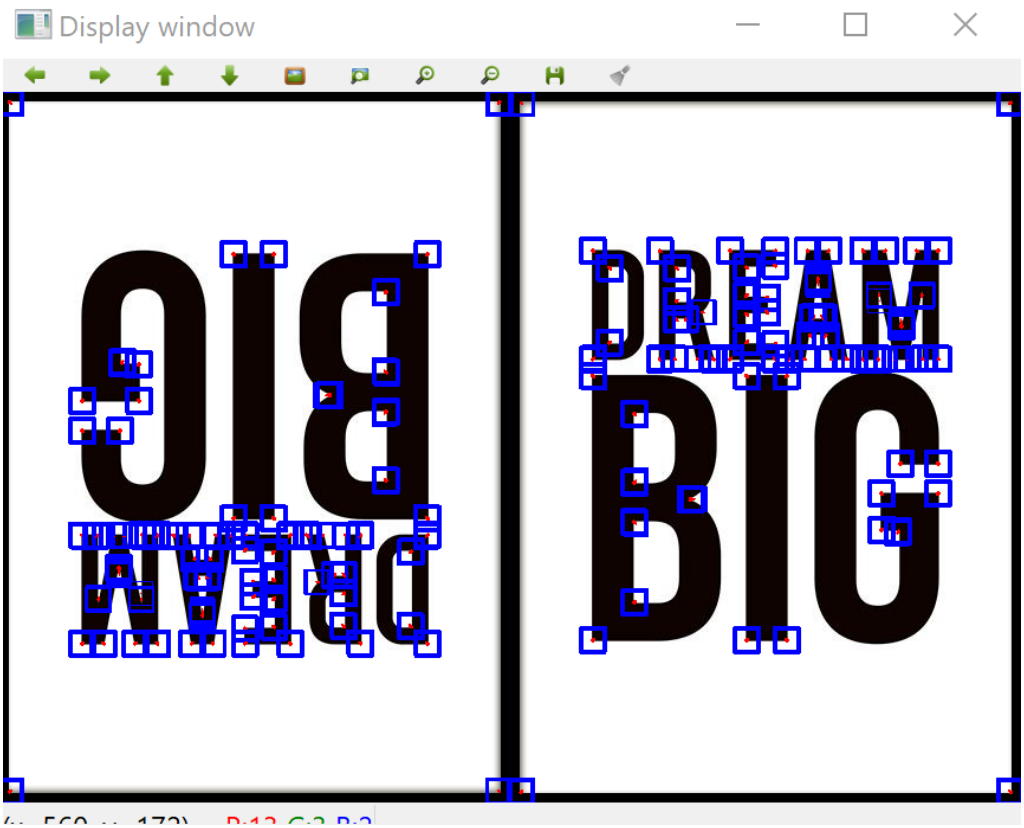
```
PS C:\Users\CHEN\Google Drive\512\cs512-f17-chen-xu\AS3\src> python AS3.py
input key to Process image(press 'H' for help, press 'q' to quit):
H
'h': Estimate image gradients and apply Harris corner detection algorithm.
'b': Obtain a better localization of each corner.
'f': Compute a feature vector for each corner were detected.
```

3. Take parameter to implement Harris:

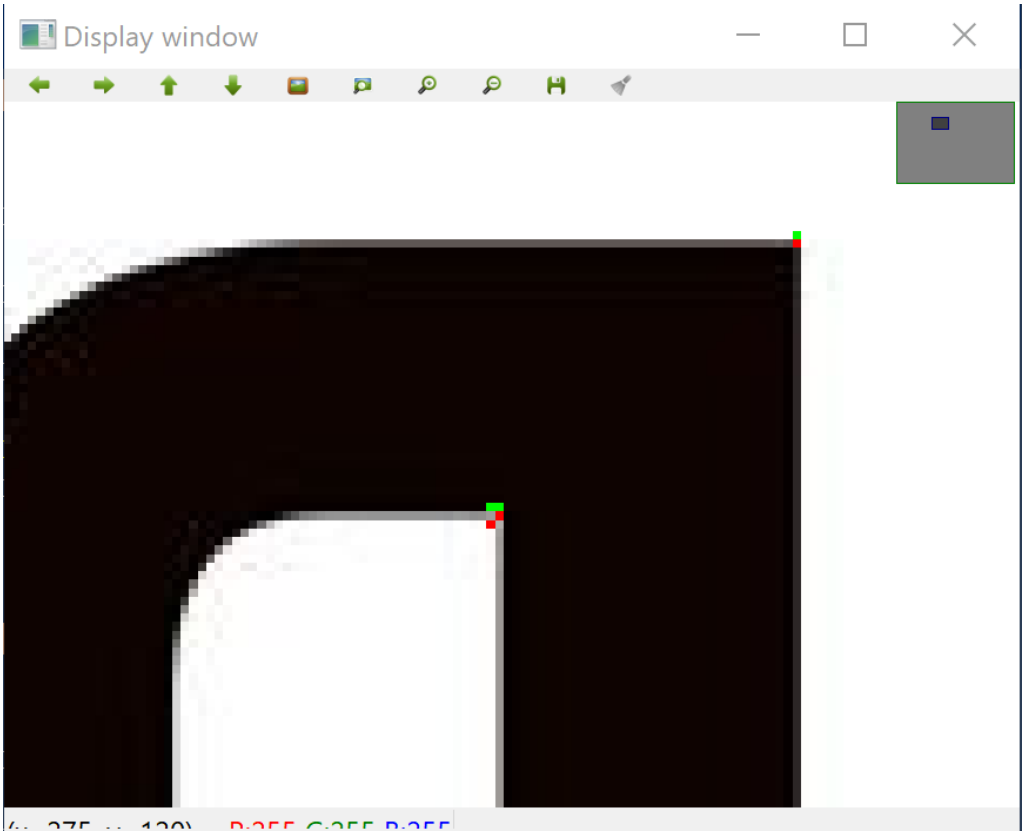
```
input key to Process image(press 'H' for help, press 'q' to quit):
h
the varience of Gussian scale(n):3
windowSize :2
the weight of the trace in the harris conner detector(k)[0, 0.5]:0.04
threshold:6000000
processing...
```

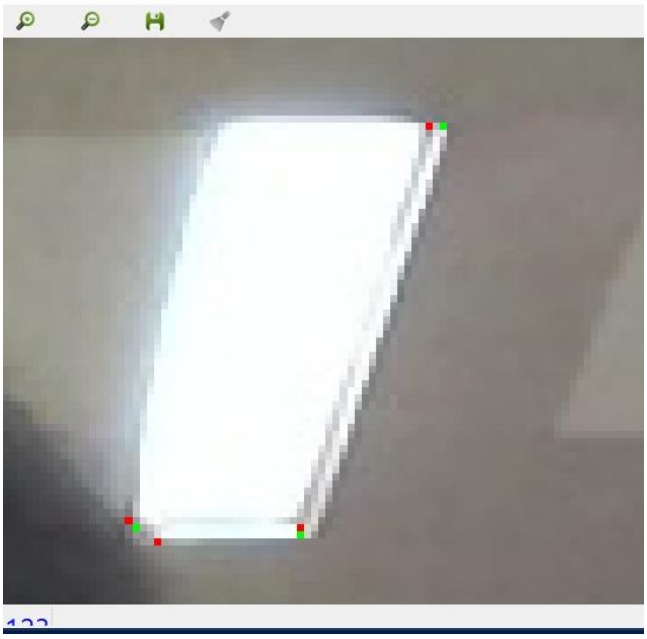
4. Get the result from Harris corner detection


Display window

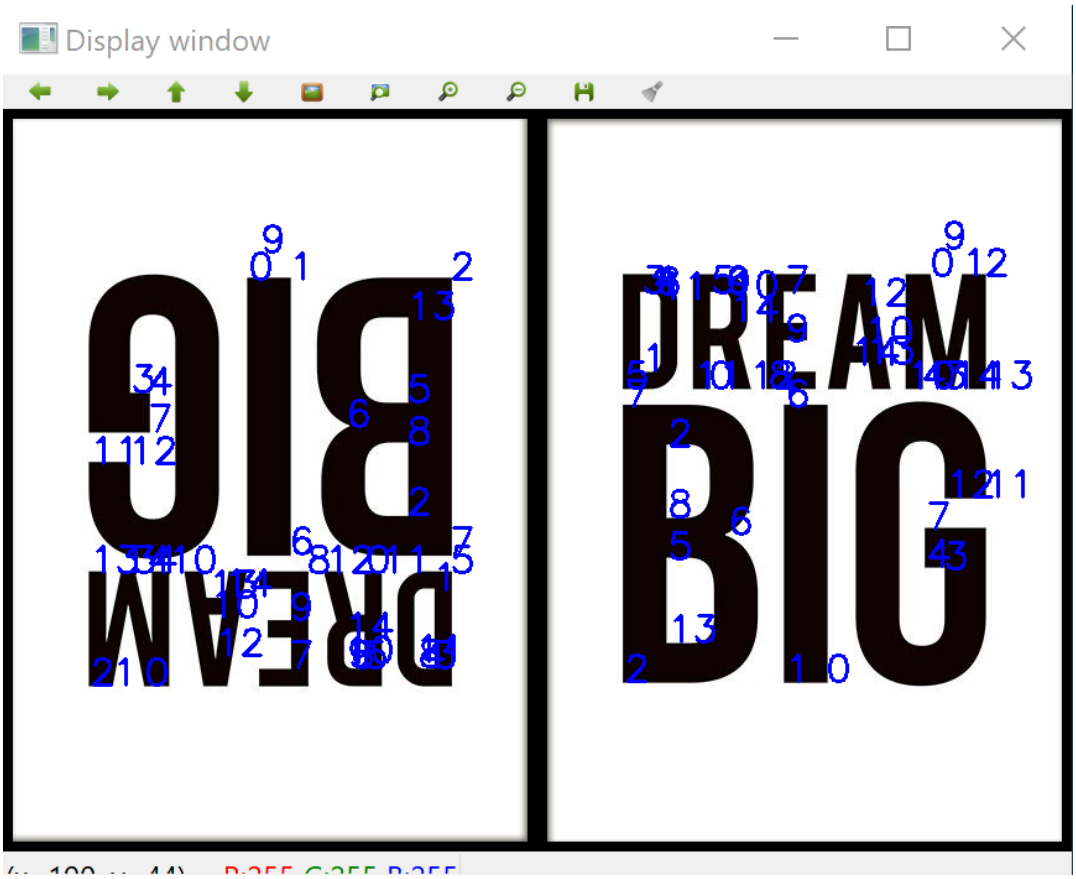

Display window
BIG
DREAM
BIG
DREAM

5. Get better location





6. Using the feature vectors, you computed match the feature points. Number corresponding points with identical numbers in the two images. I record first 50 points in each image.



5. Reference

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html

https://blackboard.iit.edu/webapps/discussionboard/do/message?action=list_messages&course_id=_63837_1&nav=discussion_board_entry&conf_id=_68225_1&forum_id=_20631_1&message_id=_171601_1