Identifying Near-duplicate Frames from Two Sets of Videos

Xu Han (xh2379@columbia.edu)

Abstract

Our goal is to identify near-duplicate frames from different videos. Nearly duplicate frames are frames that not exactly the same, but only differs in minor aspects such as the camera angle, the distance and the displayed texts, etc. Other than that, they contain almost the same content and describe the same scenes, people or events. This could associate texts from the videos and thus be useful to the further analysis. One important factor is that the process of finding such frame pairs should ideally be done automatically and involve as little manual inspections as possible.

Introduction

When we focus on videos from Chinese sources and English sources on the same topic, they usually contain frames that describe the same event or idea. Claire used a VGG-19 network in order to find near-duplicate frames. However, a great amount of manual inspection is involved. The work in this report mainly aims to improve the process and reduce the effort needed from humans while increasing the performance of finding near-duplicate frames. Below is an example of such frames.

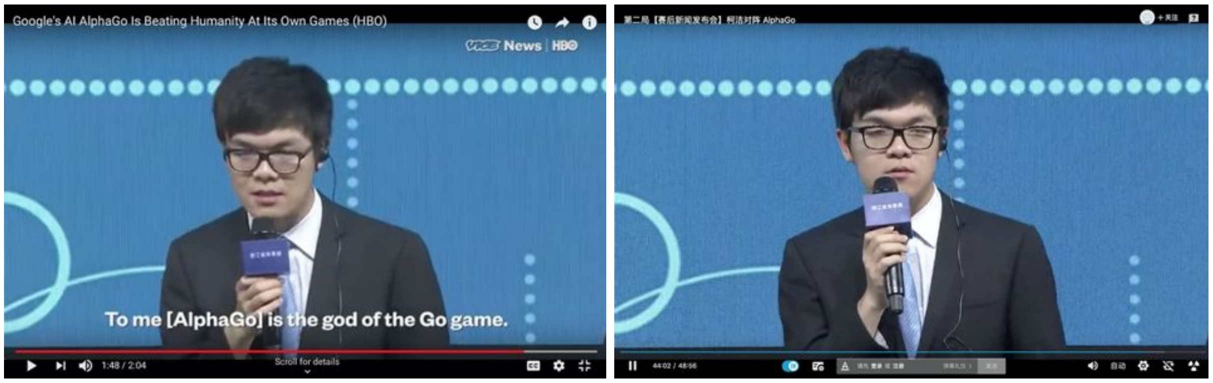

Google's AI AlphaGo Is Beating Humanity At Its Own Games (HBO)
VICE News | HBO
To me [AlphaGo] is the god of the Go game.
1:48 / 2:04
Scroll for details
第二局【赛后新闻发布会】柯洁对阵 AlphaGo
44:02 / 48:56

Figure 1. Near-duplicate frames

As we can see, the two frames contain the same background and the same person talking. The difference is the posture of him talking and the subtitles shown at the bottom.

Methods

Previous Method

In previous work, the task of finding near-duplicate frames mainly relies on manual inspection. In other words, we watched the videos and tried to see whether there are similar frames among the videos from different sources. This was obviously not very efficient, and thus needs to be improved.

Frame Sampling

To begin with the improvement, we first gather the frames from the videos. The most straightforward way to do this is sampling the videos at a constant rate. This could be easily done with the help of the open-source package, OpenCV (https://opencv.org). The following screenshot shows an example of frame sampling on a set of videos.

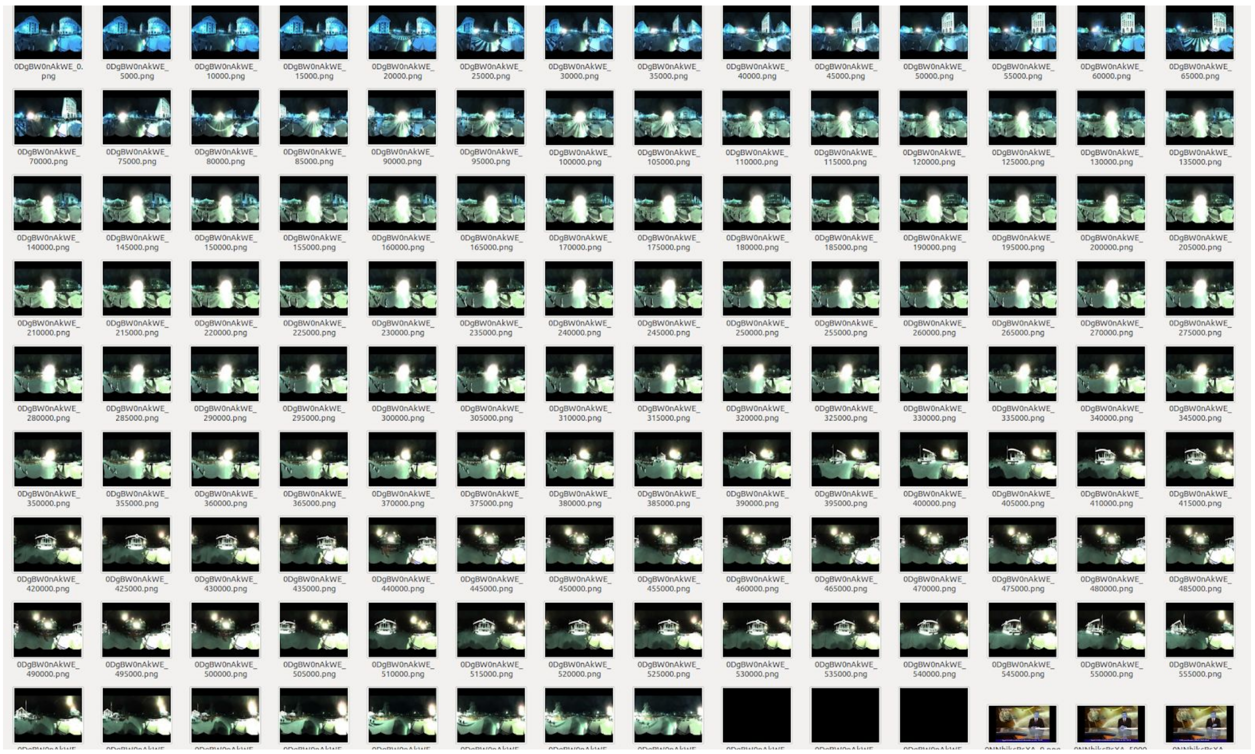


Figure 2. Screenshot of constant rate frame sampling

This breaks a video into many frames. This will of course discard some information, but in most cases key-frames will not be lost as long as the sampling rate is not too low. In the example shown the sampling rate is 5 seconds per frame.

VGG-19 Network

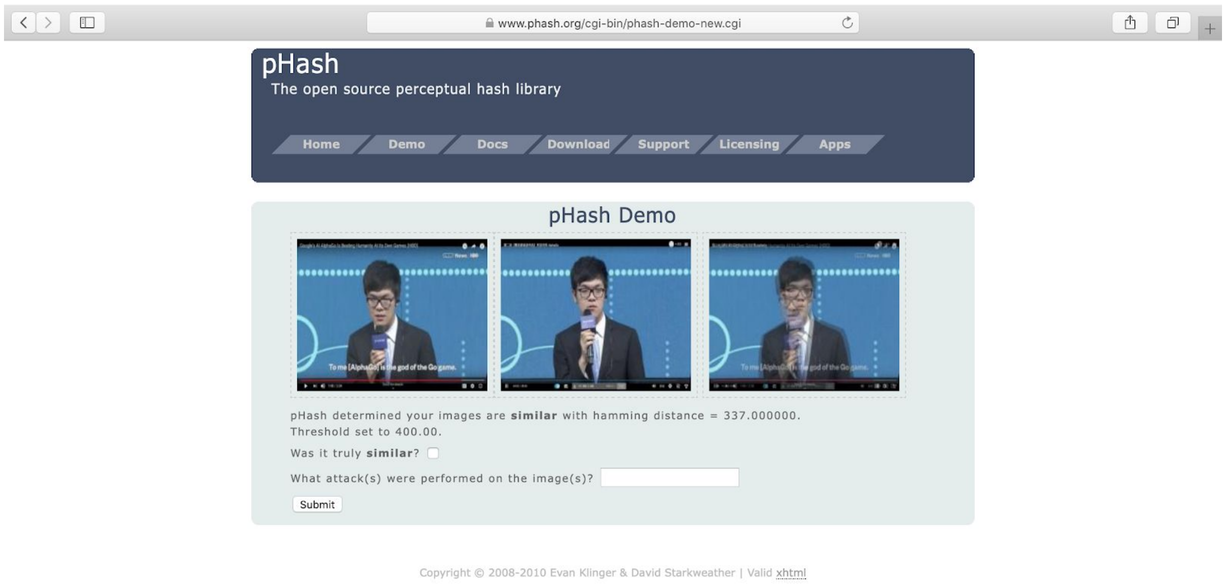
After breaking videos into individual frames, we can now focus directly on frames instead of videos. The VGG-19 network is a very handful tool in many applications working with pictures and pre-trained versions on very large datasets can be found on the web. The network includes several convolution layers and before the final output it contains several fully connected layers. In our applications we do not use the last layer's result since it would not be very meaningful as we are not working on a classification problem. Instead, we extract the output of one of the intermediate layers and treat it as a feature vector of the input frame.

With more detail, we tried the last convolution layer and the second to last layer of the entire network. The reason of choosing them is that convolution layers extract information related to the vision features of the frame, while the fully connected layers are believed to relate to conceptual features more.

Hashing

Besides VGG-19, we also tried another method to "featurize" a frame, which is hashing. Unlike cryptographic hashing, which gives substantially different results even if the input differs by a tiny bit, the hashing we would expect here has to give similar outputs whenever the input is similar. Then we can tell from the results whether two frames are similar.

We use a Python package named ImageHash (https://pypi.org/project/ImageHash/) to complete this. It outputs a number for each frame, which can be used to compare frames. A demo of pHash (perceptual hashing) can be found at https://www.phash.org/demo/ and is shown below.


www.phash.org/cgi-bin/phash-demo-new.cgi
pHash
The open source perceptual hash library
Home
Demo
Docs
Download
Support
Licensing
Apps
pHash Demo
Submit

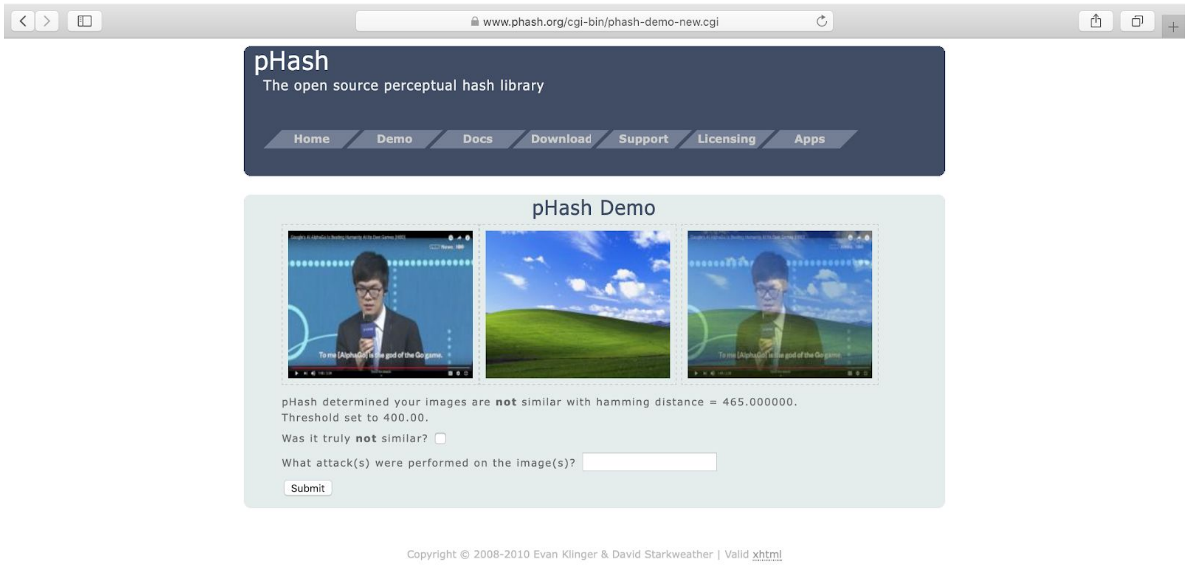
pHash determined your images are **similar** with hamming distance = 337.000000.
Threshold set to 400.00.

Was it truly **similar**?

What attack(s) were performed on the image(s)?

Copyright © 2008-2010 Evan Klinger & David Starkweather | Valid xhtml

Figure 3. Demo of pHash using similar frames


www.phash.org/cgi-bin/phash-demo-new.cgi
pHash
The open source perceptual hash library
Home
Demo
Docs
Download
Support
Licensing
Apps
pHash Demo
Submit

pHash determined your images are **not** similar with hamming distance = 465.000000.
Threshold set to 400.00.

Was it truly **not** similar?

What attack(s) were performed on the image(s)?

Copyright © 2008-2010 Evan Klinger & David Starkweather | Valid xhtml

Figure 4. Demo of pHash using very different frames

Distance Function

Using either VGG-19 or hashing, we have converted frames into a vector or a number. The next step is then specify a way to determine whether two frames are similar. For this step, we tried several distance functions to determine how similar two frames are.

L1 Distance: This is the simplest distance function we used. For two vectors, take the sum of the absolute differences of each entry. For a single number, simply take the absolute difference between them. This is the only distance function among the ones used for hashing.

L2 Distance: Instead of taking the sum of absolute differences of each entry, take the sum of the square of the absolute differences.

Cosine Angular Distance: Using the Euclidean dot product formula, one can calculate the angle between two vectors. The cosine of that angle is then defined as the angular distance. One advantage of this method is that the result is on a normalized scale.

Removing Similar Frames within the Same Set

From the screenshot shown in Figure 2, it is not too difficult to see that a large number of similar frames exist in the result of frame sampling. This is due to the fact that the video contains almost the same content in a period of time. Frame sampling will capture many copies of those contents. This will cause many duplications in our final results. Thus before trying to capture near-duplicate frames from two sets of videos, we first need to eliminate similar frames within the same set. We use average hashing (http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html), which is a simple hashing method based on pixels to remove such similar frames within the same set of videos. Whenever the result of aHash of a frame has a L1 distance less than some threshold to another frame within the same set, the frame is discarded. The result after this step is shown below.

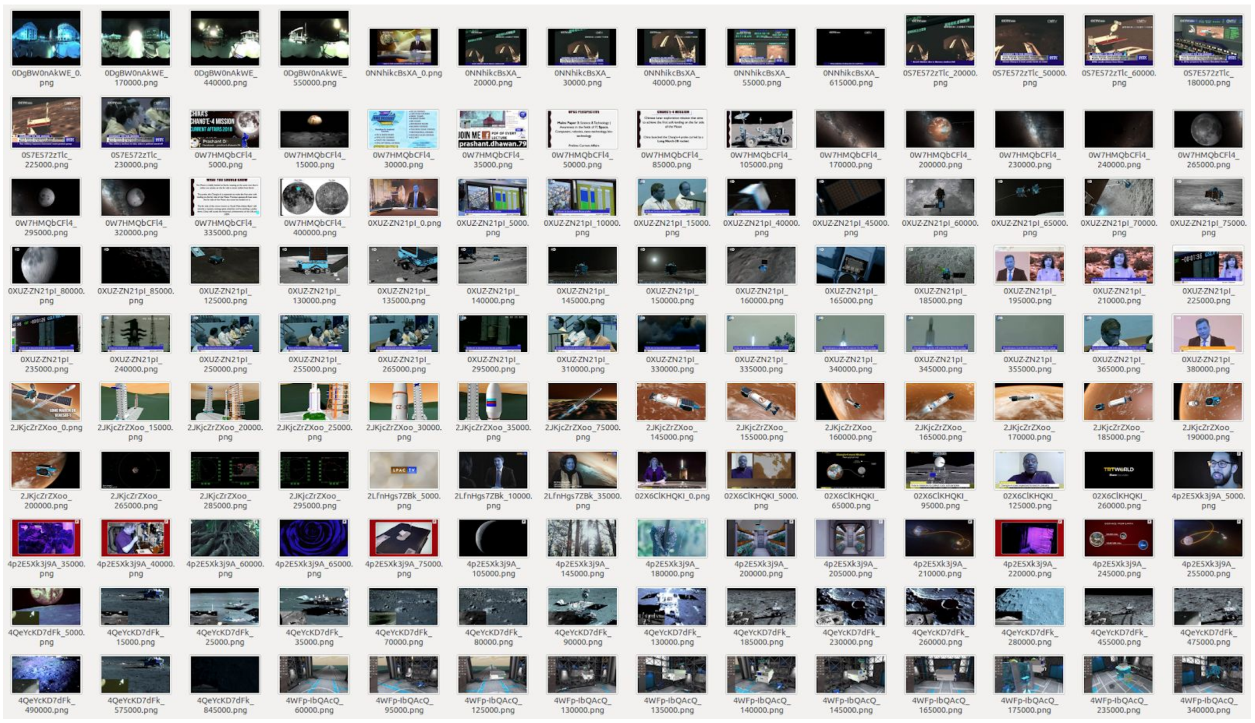


Figure 5. Screenshot of frame sampling after removing similar frames

As we can see, this simple method turned out to be quite effective, and it greatly reduced the number of frames in a set of videos, enhancing the performance of the model.

Finding Near-duplicate Frames from Two Sets of Videos

After setting up all the previous tools, we can finally start to actually work on finding the near-duplicate frames. For a single topic, we gather a set of Chinese videos and a set of English videos. For each set of videos, sample the frames at a constant rate and remove duplicate ones using aHash with L1 distance. Then using VGG-19 with the last convolution layer or second last fully connected layer, or using hashing, convert the frames into vectors or numbers. Finally, compare each frame in one set and each frame in the other and rank the candidate frame pairs according to a distance function. Part of the result is shown below to demonstrate what it looks like.

```
["av1007072", 0, "4QeYcKD7dFk", 845000, 0.03997420386806122],

["av1007072", 0, "0W7HMQbCFl4", 240000, 0.04034107040385553],

["av23040459", 220000, "4QeYcKD7dFk", 845000, 0.05797681805467925],

["av23040459", 220000, "0W7HMQbCFl4", 240000, 0.06383399460236752],
```

["av6174767", 420000, "0W7HMQbCFl4", 240000, 0.06935971250464631],

["av1007072", 0, "02X6ClKHQKI", 260000, 0.07072534716776036],

["av6174767", 420000, "4QeYcKD7dFk", 845000, 0.07147299200659595],

["av23040459", 230000, "4QeYcKD7dFk", 845000, 0.07166437959896402],

["av37679601", 45000, "4WV-NGcMMLA", 245000, 0.07228764384710153],

["av23040459", 110000, "4QeYcKD7dFk", 845000, 0.07290208102437418],

["av23040459", 230000, "0W7HMQbCFl4", 240000, 0.07404724761609144],

["av23040459", 220000, "0W7HMQbCFl4", 170000, 0.07511728212693856],

["av23040459", 110000, "0W7HMQbCFl4", 240000, 0.07587277694013275],

["av1007072", 0, "0W7HMQbCFl4", 170000, 0.07826463803966736],

["av30883884", 35000, "4QeYcKD7dFk", 845000, 0.07929353963351285],

["av1007072", 0, "0W7HMQbCFl4", 230000, 0.08031203941843484],

["av23040459", 225000, "4QeYcKD7dFk", 845000, 0.08230764993918332],

["av30883884", 35000, "0W7HMQbCFl4", 240000, 0.08392521897574544],

["av23040459", 230000, "0W7HMQbCFl4", 170000, 0.08475736368962533],

["av1007072", 0, "8WLZ8PHBnx8", 140000, 0.08516623588881521],

["av23040459", 220000, "0W7HMQbCFl4", 230000, 0.08576342209524666],

["av23040459", 225000, "9OlB99lQtPo", 265000, 0.08579177686667606],

["av1007072", 0, "4WV-NGcMMLA", 5000, 0.08600741163059768],

["av23040459", 110000, "0W7HMQbCFl4", 230000, 0.08637908775793253],
["av6174767", 420000, "0W7HMQbCFl4", 230000, 0.08670055250849396],
["av23040459", 110000, "0W7HMQbCFl4", 170000, 0.08670147268675714],
["av1007072", 0, "9OlB99lQtPo", 5000, 0.08674405701895739], ["av6174767",
420000, "0W7HMQbCFl4", 170000, 0.08815459543267735], ["av1007072", 0,
"5LbolYeLlGk", 35000, 0.08824427961102915], ["av1007072", 0,
"2LfnHgs7ZBk", 5000, 0.08826902007402276], ["av1007072", 0,
"8WLZ8PHBnx8", 85000, 0.08955250748280663], ["av23040459", 225000,
"0W7HMQbCFl4", 240000, 0.0895954143517177], ["av23040459", 110000,
"02X6ClKHQKI", 260000, 0.09019344484542155], ["av6174767", 420000,
"02X6ClKHQKI", 260000, 0.09083248493326701], ["av23040459", 110000,
"4WV-NGcMMLA", 5000, 0.09088503944438073], ["av23040459", 220000,
"9OlB99lQtPo", 265000, 0.09133901935121425], ["av6174767", 420000,
"4WV-NGcMMLA", 5000, 0.09145786464277246], ["av1007072", 0,
"0XUZ-ZN21pI", 85000, 0.09150083791630045], ["av23040459", 15000,
"4QeYcKD7dFk", 845000, 0.0917947542372095], ["av23040459", 220000,
"02X6ClKHQKI", 260000, 0.09182617310739155], ["av23040459", 15000,
"0W7HMQbCFl4", 240000, 0.09192202342874406], ["av23040459", 220000,
"4WV-NGcMMLA", 5000, 0.09196132547559317]

The entries stand for the Chinese video identifier, the frame identifier, the English video identifier, the frame identifier and the distance. The first few pairs are shown below. The left column are frames from Chinese videos and the right column from English. Each row is a candidate near-duplicate pair.

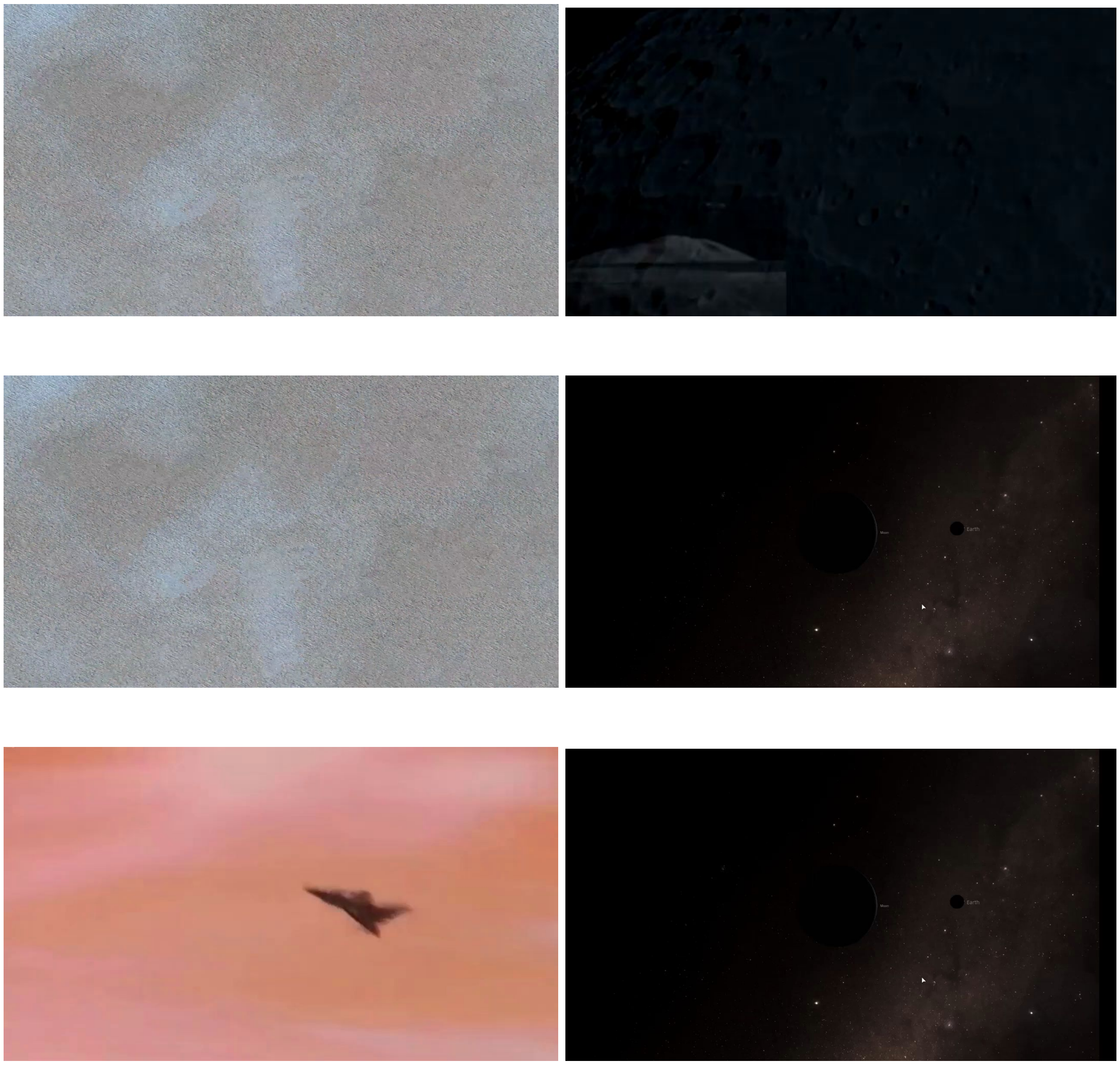
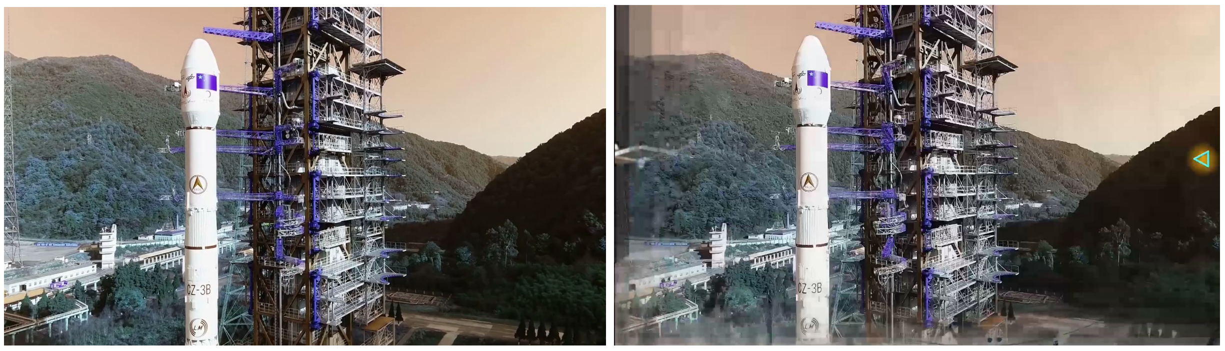
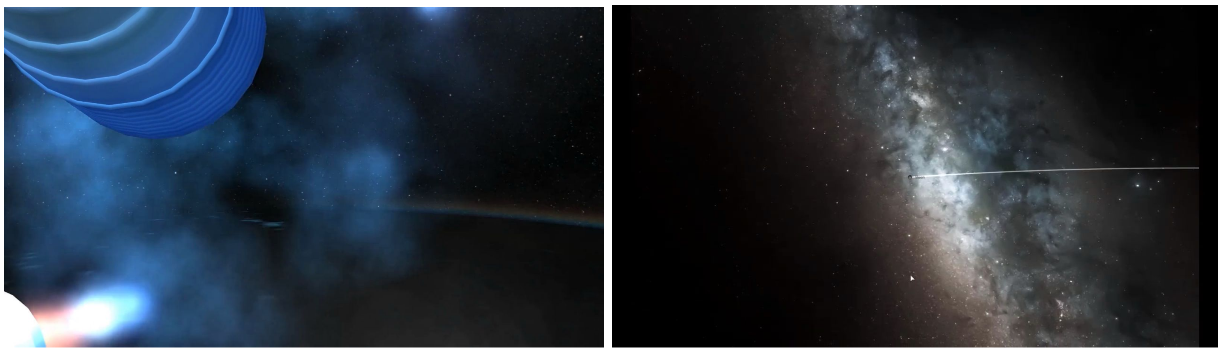


Figure 6. Top 5 near-duplicate candidate pairs

Removing Noise

From the results above, we see that the pairs found by the process are not very interesting. Some frames are easily matched with others and contain little useful information. Currently, this problem is solved by manually removing additional frames from the sets of videos to be processed and could be improved in future work. The result after such manual noise removal is shown in the following.





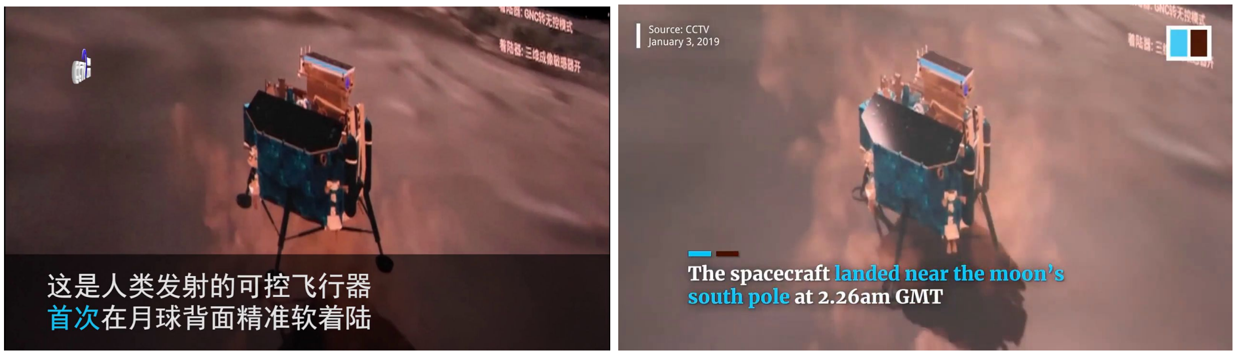


Figure 7. Top 3 near-duplicate candidate pairs after noise removal

Comparisons

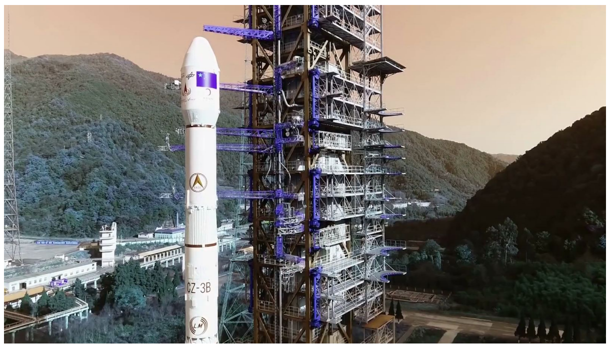
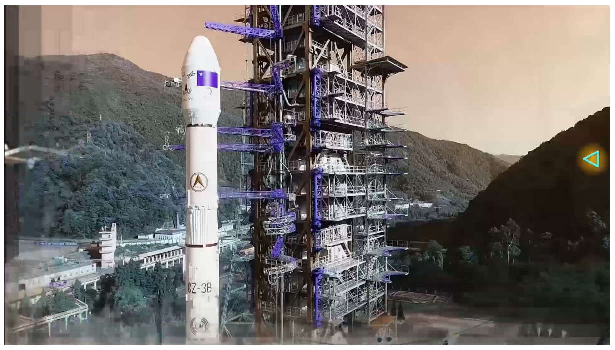
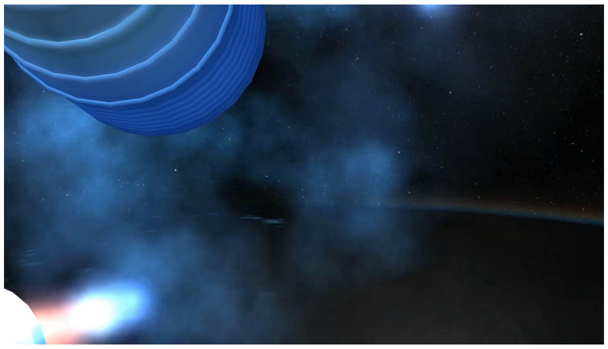
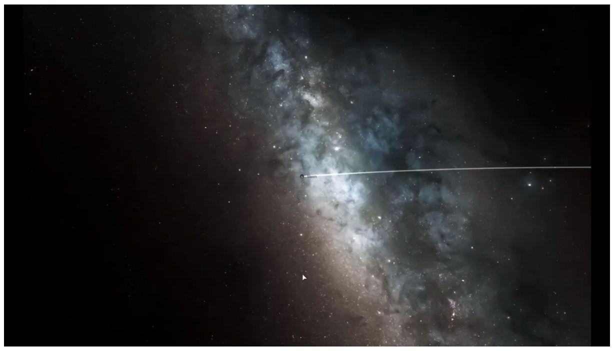
In the previous sections, there are many choices left in the process of finding near-duplicate frames. When converting frames into vectors or numbers, we could use the last convolution layer of VGG-19 or its second to last fully connected layer. We could also use hashing. When comparing the frames to determine whether they are similar, we have several choices of distance functions. In the previous results, we used the last convolution layer of VGG-19 and angular distance. In this section, comparisons will be made to see the difference between different choices.

Distance Functions

While keeping other factors fixed (the last convolution layer of VGG-19), we compare the results of choosing L1 distance, L2 distance and angular distance.

Angular Distance

The result of using angular distance is shown below.

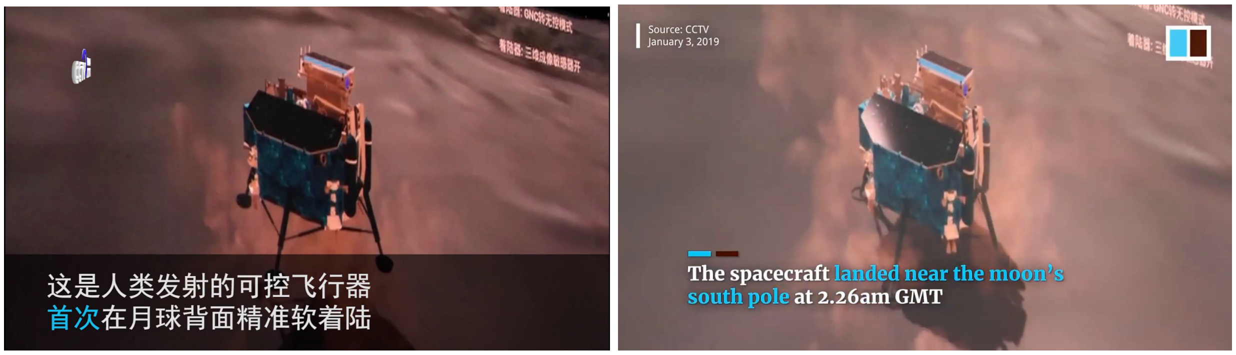
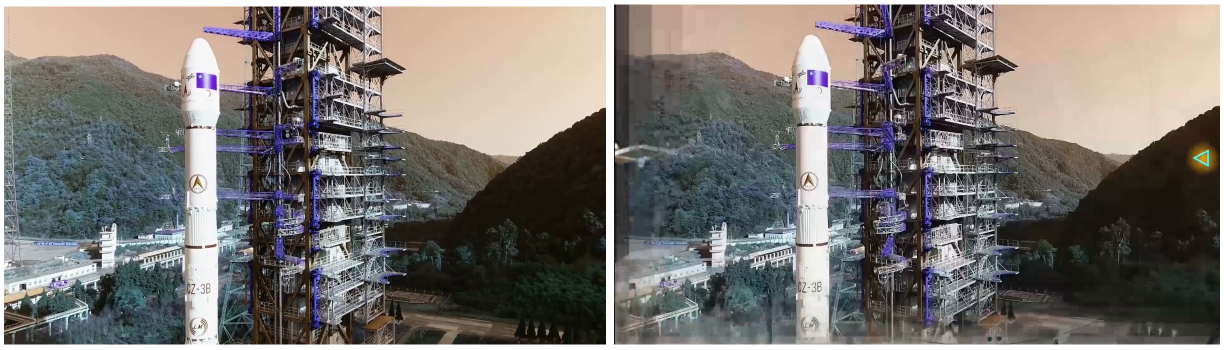
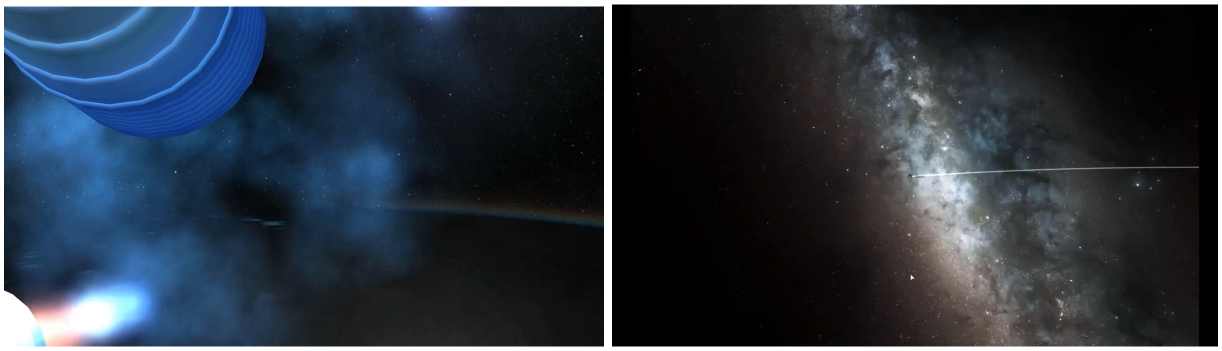


Figure 8. Top 3 near-duplicate candidate pairs using angular distance

L1 Distance

The result of using L1 distance is shown below.





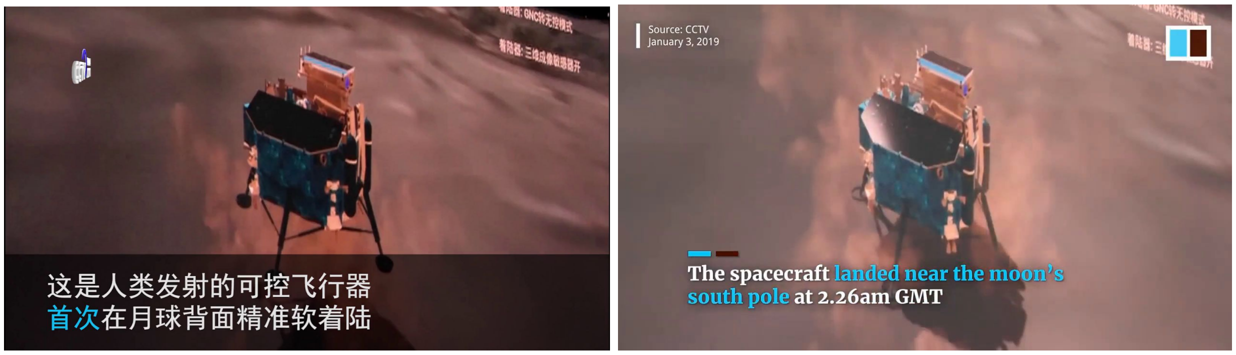
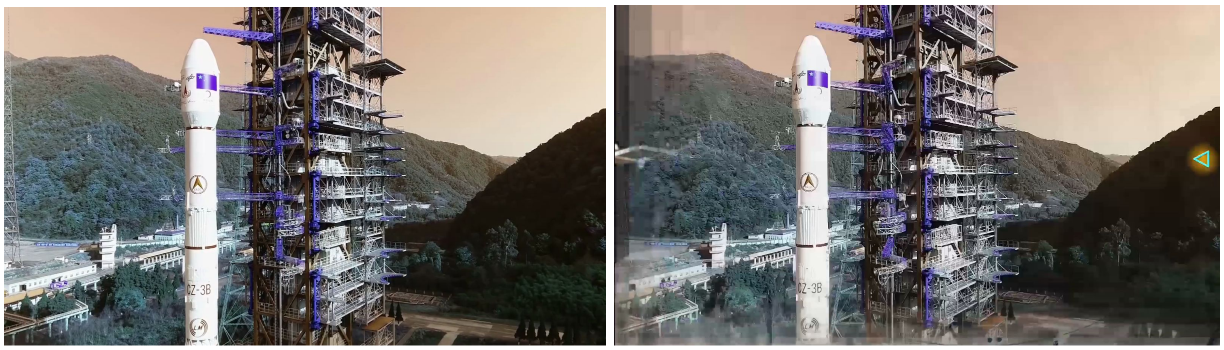
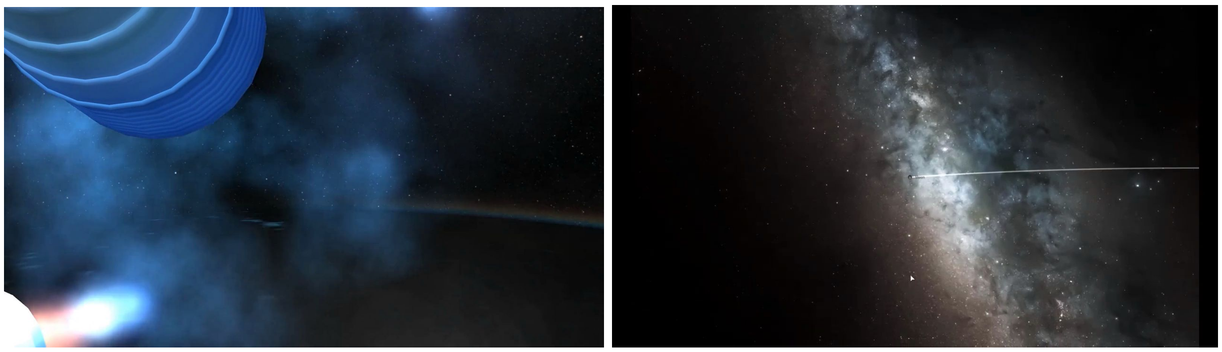


Figure 9. Top 3 near-duplicate candidate pairs using L1 distance

L2 Distance

The result of using L2 distance is shown below.





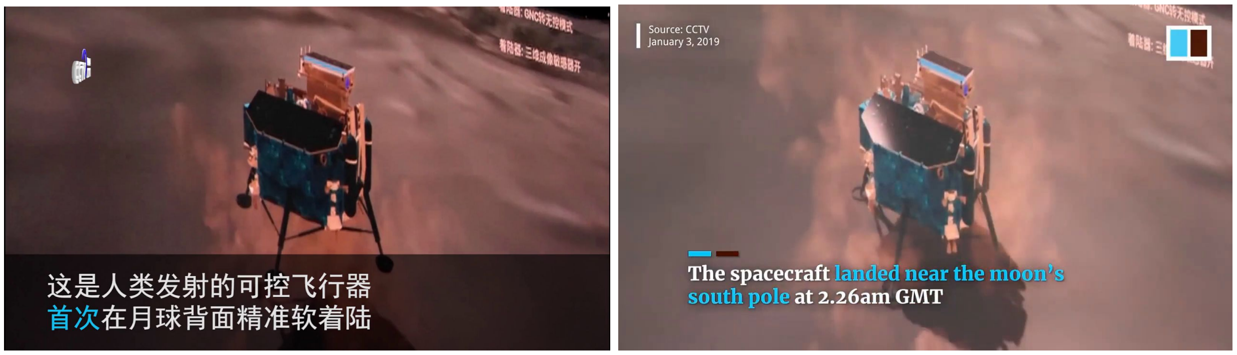


Figure 10. Top 3 near-duplicate candidate pairs using L2 distance

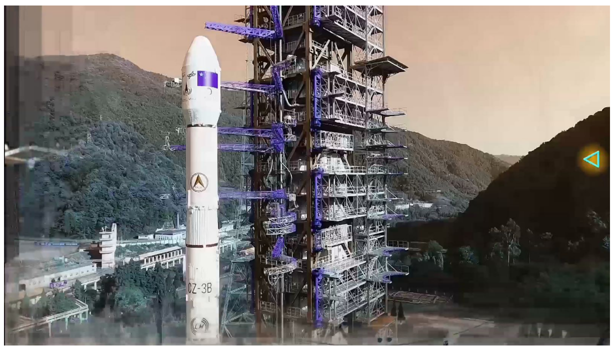
From the results above and the results of several other sets of videos, we conclude that the choice of distance function does not have a large impact on the results, and therefore we can choose any one of them or our purpose. We chose angular distance since it is naturally normalized.

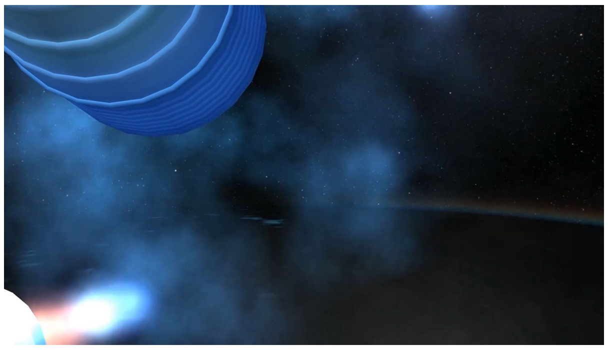
Method of Featurizing Frames

We now compare the effect of choosing different methods of converting frames into vectors or numbers. Since only L1 distance can be applied on hashing and the choice of distance function does not matter much, we choose L1 distance as the distance function for the purpose of this comparison.

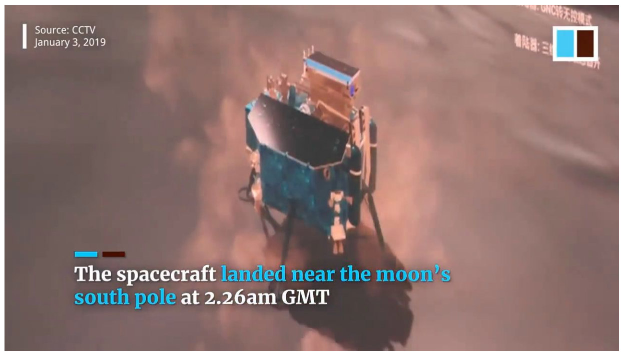
VGG-19 Last Convolution Layer

The result of using the last convolution layer of VGG-19 is shown below.





这是人类发射的可控飞行器
首次在月球背面精准软着陆


Source: CCTV
January 3, 2019
The spacecraft landed near the moon's south pole at 2.26am GMT

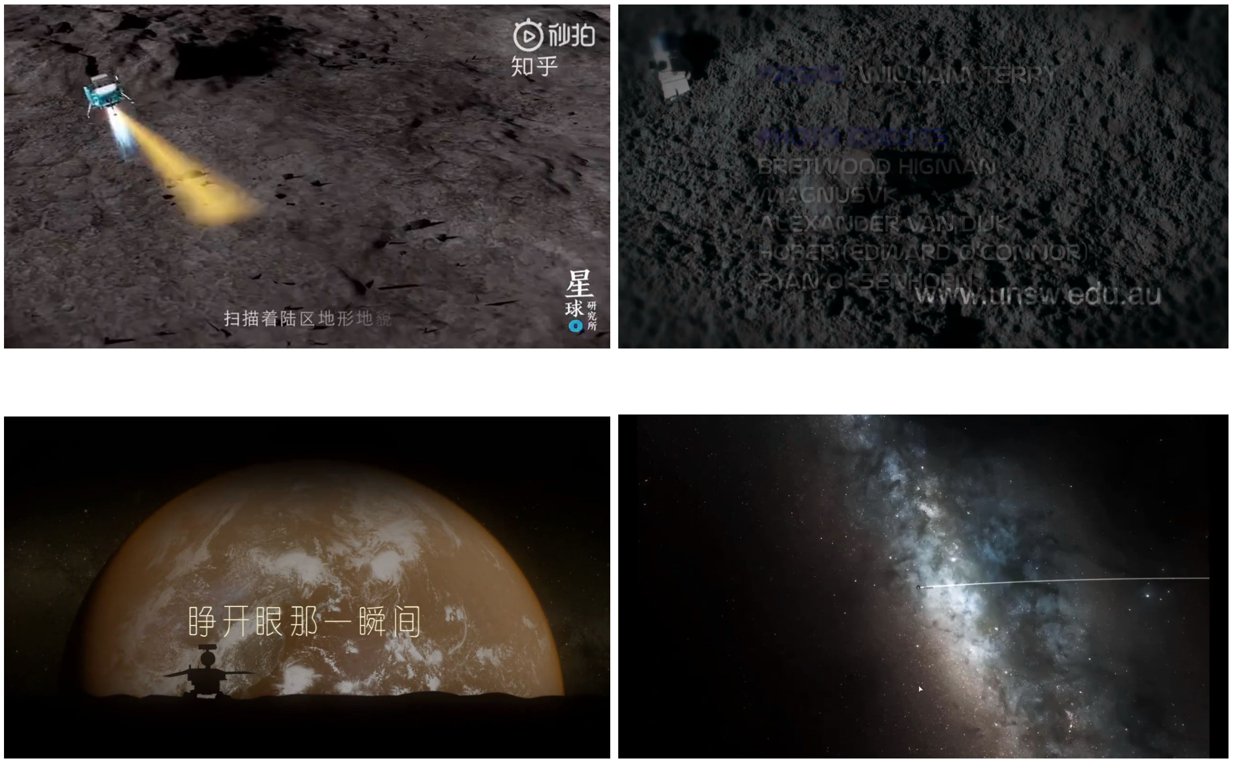
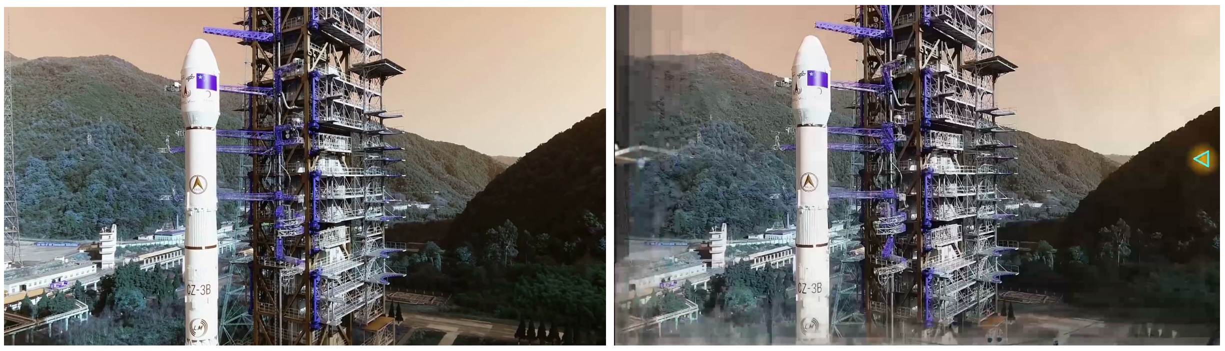


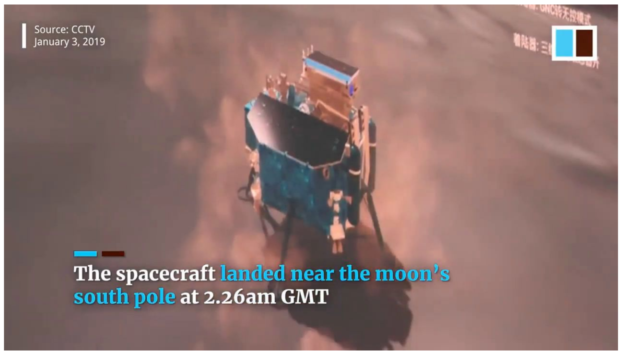
Figure 11. Top 5 near-duplicate candidate pairs using the last convolution layer of VGG-19

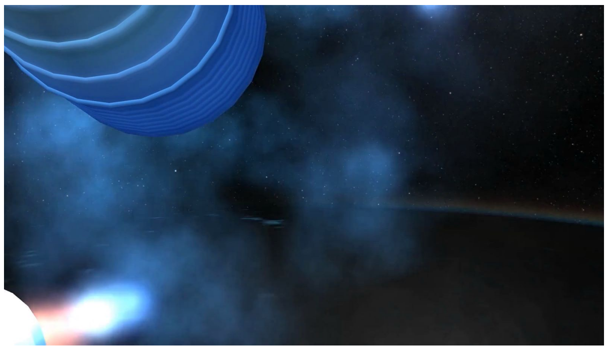
VGG-19 Second to Last Fully Connected Layer

The result of using the second to last fully connected layer is shown below.

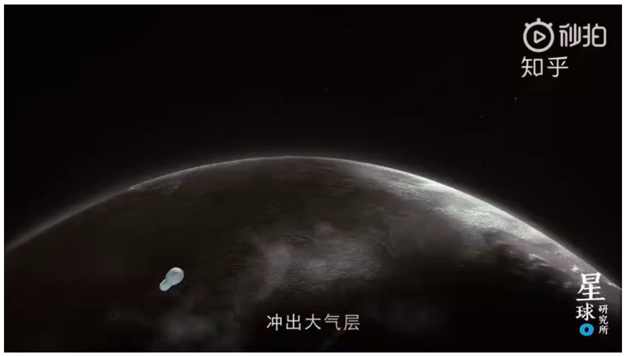


着陆器: GNC转无控模式
着陆器: 三维成像敏感器开
这是人类发射的可控飞行器
首次在月球背面精准软着陆


Source: CCTV
January 3, 2019
The spacecraft landed near the moon's south pole at 2.26am GMT



高清


秒拍
知乎
冲出大气层
星球研究所

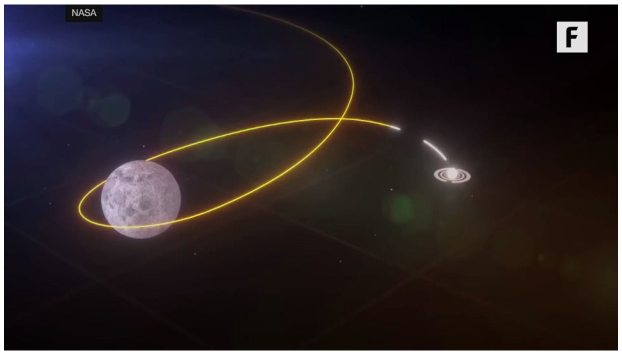
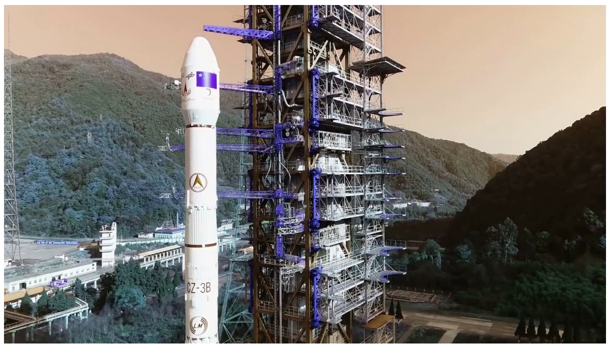
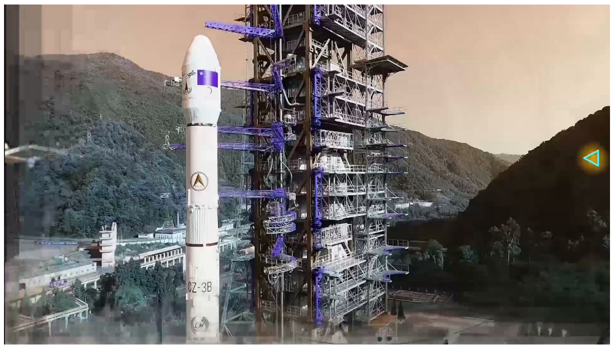

NASA
F

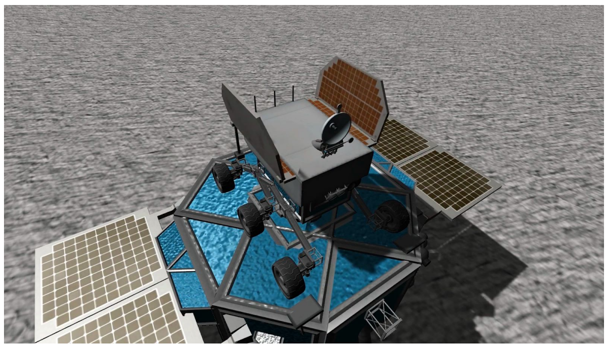
Figure 12. Top 5 near-duplicate candidate pairs using the second to last fully connected layer of VGG-19

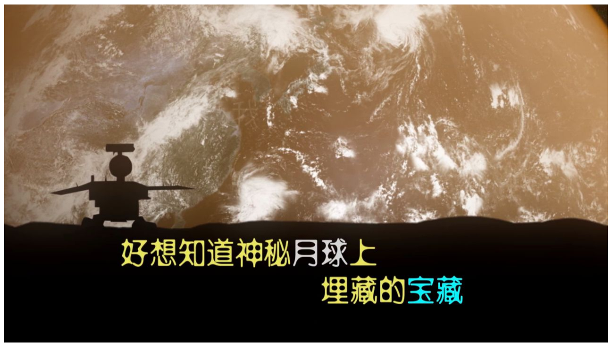
aHash

The result of using aHash is shown below.







TRT
skype
MOON MISSION
Chang'e-4 rover expected to land in January
NEWS
RUSSIAN FOREIGN MINISTER LAVROV SAYS KREMLIN WILL ONLY DISCUSS


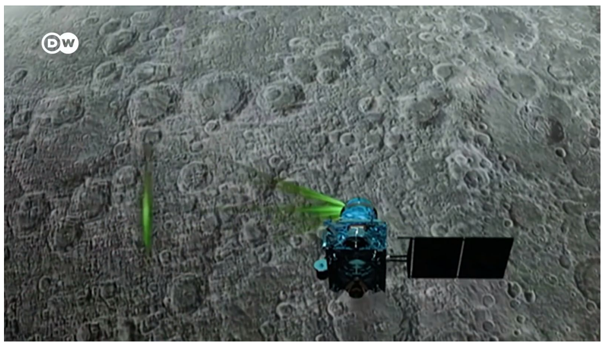


Figure 13. Top 3 near-duplicate candidate pairs using aHash

pHash

The result of using pHash is shown below.

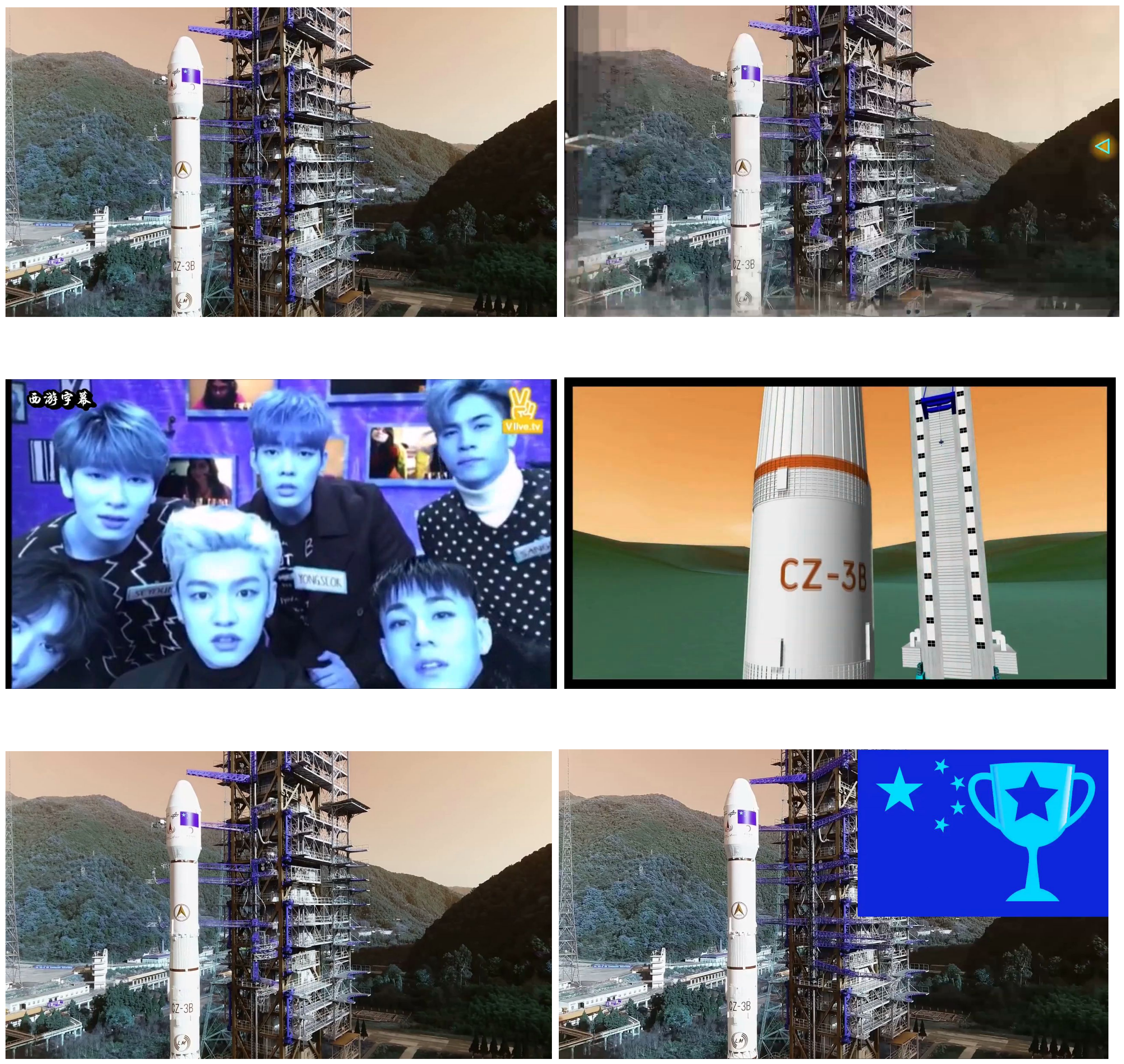


Figure 14. Top 3 near-duplicate candidate pairs using pHash

dHash

The result of using dHash is shown below.

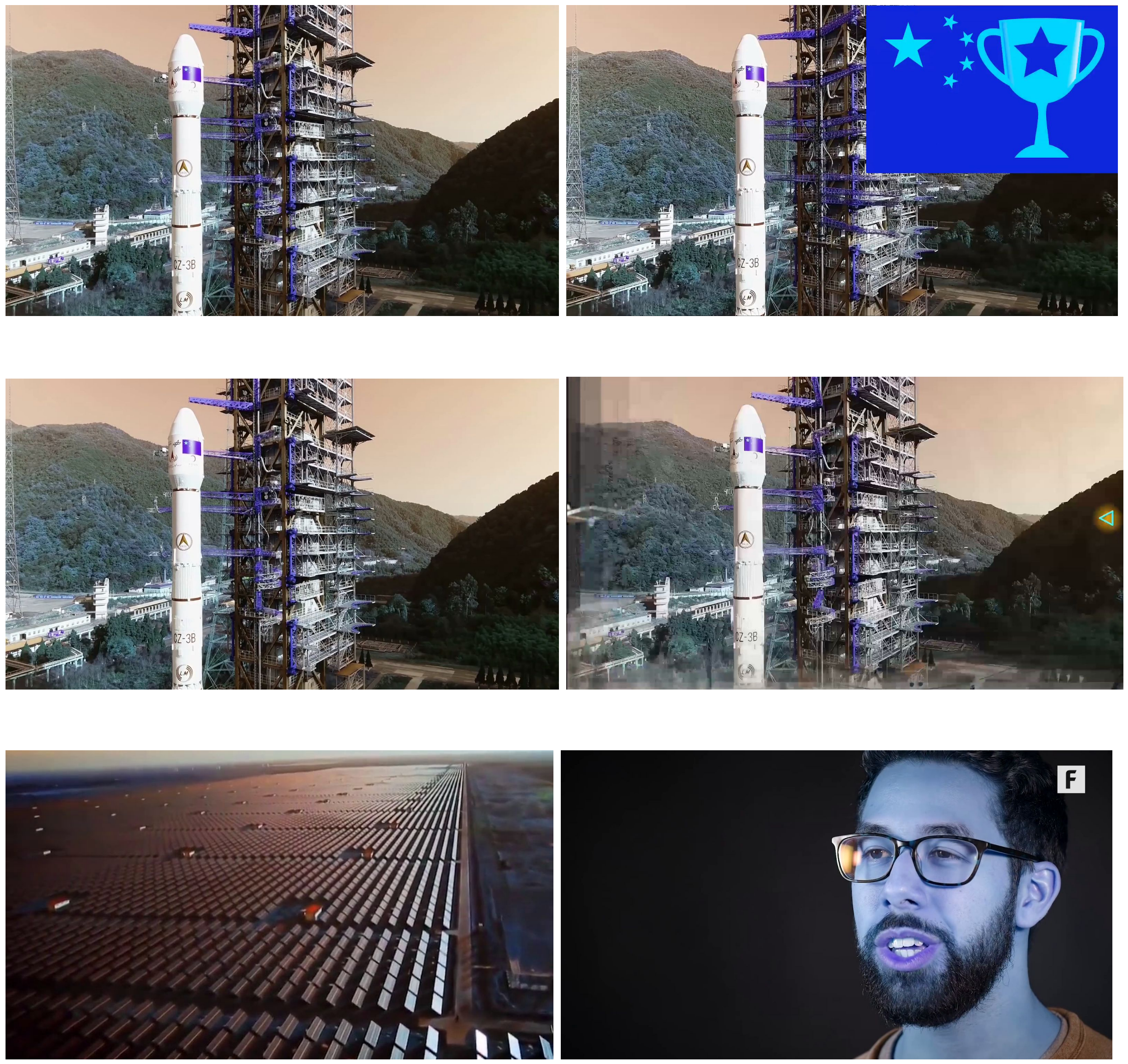


Figure 15. Top 3 near-duplicate candidate pairs using dHash

wHash

The result of using wHash is shown below.

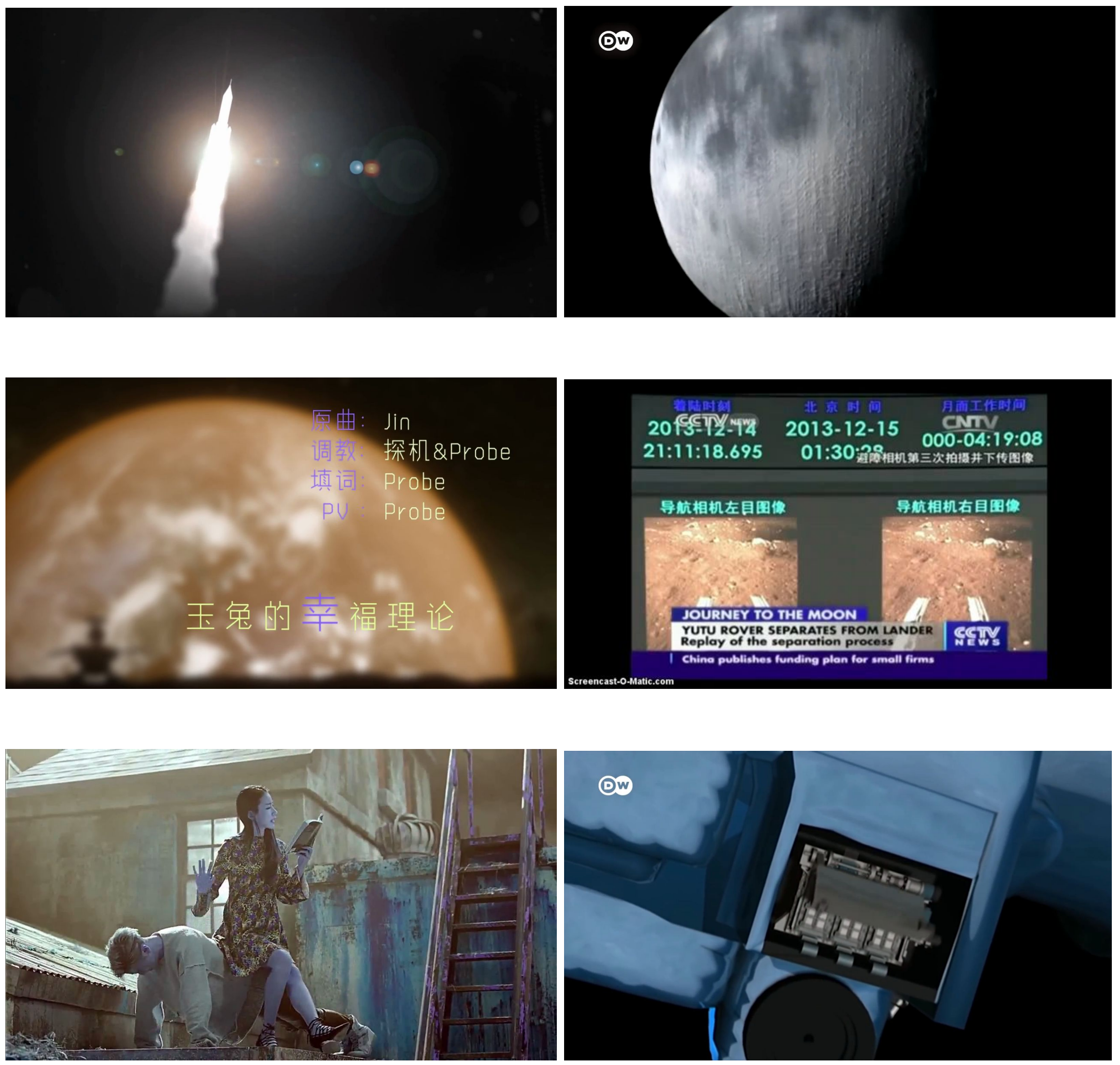


Figure 16. Top 3 near-duplicate candidate pairs using wHash

From the results above, we see that VGG-19 performs better than hashing, while using the last convolution layer or using the second to last fully connected layer have similar performance. In other data we have tested, the results are similar, and using the last convolution layer yields slightly better results.

Possible Improvements

Although the process shown greatly increased performance and efficiency compared to previous methods, it still has many downsides and some of the improvements that should be made in future work.

Removing Noise

Using aHash successfully removed most of the similar frames within the same set of videos. However, the frames that contain very little information but are very easy to match other frames, such as frames totally black, can greatly harm the results. This ideally should also be automated, but now it is done by manual inspection and should be improved.

Result Format

Currently, the format of the results is in json. The results are then manually inspected and the corresponding frames are then observed. This is very inefficient. It will be much more efficient if a better interface is built and used to observe the results.

Conclusion

In short, we developed an automated process to identify near-duplicate frames from two different set of videos. The process is summarized in the following:

1. Download a set of Chinese videos and a set of English videos on the same topic.
2. Perform frame sampling at a constant rate
3. Remove similar frames using aHash within the same set of videos.
4. Remove frames that contain little information.
5. Choose a method of featurizing frames and a distance function.
6. Calculate the distance between each frame of one set and each frame of the other set.
7. Rank the candidate pairs of frames and observe the top few pairs.