Image shuffle

Faisal Qureshi
Professor
Faculty of Science
Ontario Tech University
Oshawa ON Canada
http://vclab.science.ontariotechu.ca (http://vclab.science.ontariotechu.ca)

Copyright information

License

Outline

- Numpy arrays and OpenCV images

In [1]:
```
import cv2 as cv
import numpy as np
import scipy as sp
from scipy import signal
import matplotlib.pyplot as plt
```

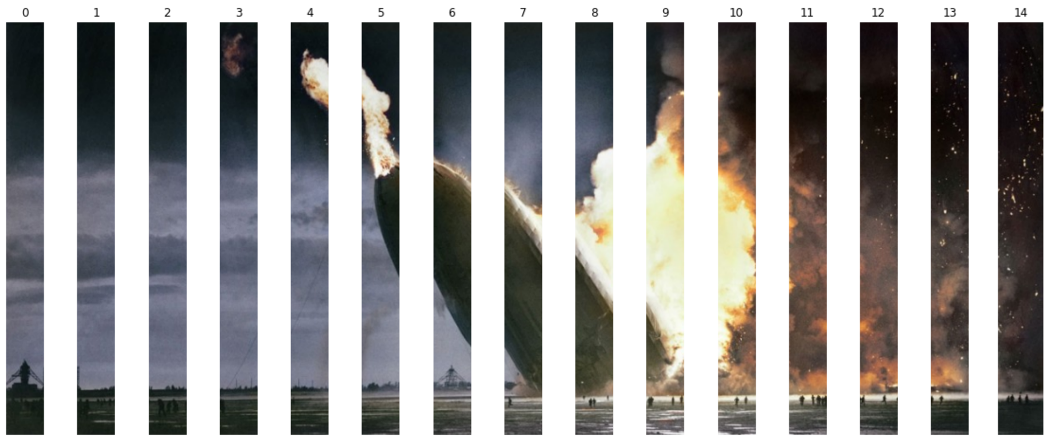
In [2]:
```python
img = cv.imread('data/hindenburg.jpg')
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)

plt.figure(figsize=(10,10))
plt.title('Hindenburg')
plt.imshow(img)
plt.xticks([])
plt.yticks([]);
```

Hindenburg

Task 1: Cut image into strips

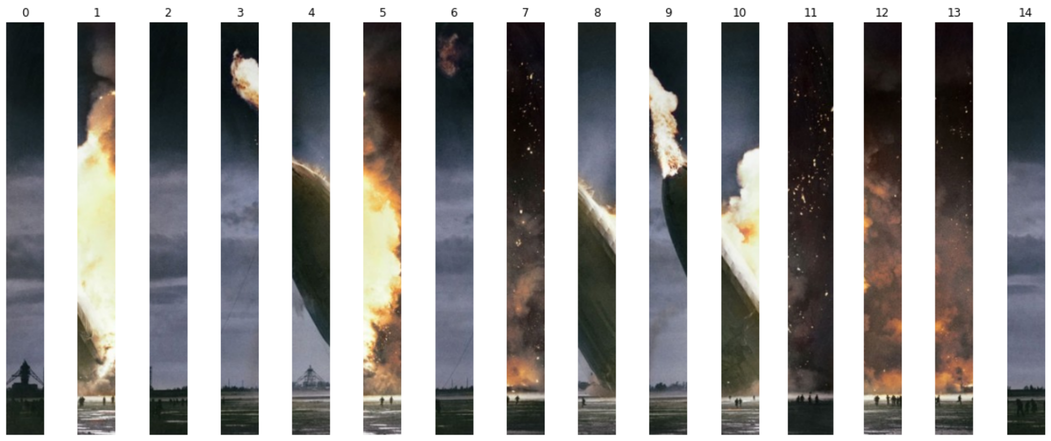
Below, we have cut the image into 15 strips, indexed from 0 to 14.



In [4]:
```
# Your work goes here
```

Task 2: Shuffle the strips

Shuffle the image strips that you have constructed in the previous task. As shown below.



In [5]:
```
# Your work goes here
```

Task 3: Reconstruct the image

Now reconstruct the original given a shuffled list of strips.

Solution sketch

In order to reconstruct the original image, you'll need to examine the boundary pixels of various patches. Patches whose boundary pixels "match scores" are high are more likely to be neighbouring patches. One way to compute match scores is to treat each boundary as a real-valued vector. Then the problem is simply to compute the similarity between two vectors. I encourage you to use Euclidean distance and Cosine similarity to see if two vectors (i.e., boundaries) are similar.

Mathematically, say $\mathbf{x}_i$ and $\mathbf{x}_j$ are two vectors representing boundaries of two patches. We can compute the similarity between these two vectors as follows.

Euclidean distance

$$\text{Euclidean-distance}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sqrt{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}$$

Cosine similarity

$$\text{Cosine-similarity}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$$

Concerns about efficiency

I encourage you to think of ways on how to reconstruct the image efficiently.

What to submit

You need to submit the following:

- Your solution and the resulting image;
- An analysis of which similarity scheme works best for this image; and
- What techniques did you employ to speed up the process.

In [7]:
```
# Your work goes here
```

Task 4 (Optional): Divide the images in rectangles and try to reconstruct it.

In [6]:
```
# Your work goes here
```

OntarioTech UNIVERSITY

```
In [ ]:
```