# *Hybrid Images*

**Edgar Andrés Margffoy Tuay** 201412566
Universidad de los Andes Biomedical Engineering
Computer Vision 23 de febrero de 2017

**Resumen**

Hybrid images is a method to merge two images, based on their frequency components and the biological process of image formation on the human eye, which disntiguishes low and higher frequency images based on the distance between the eye and the image, this approach allows to combine to images into one and trick the human eye to see one of the images when the composite is close, and the second image otherwise.

## 1. Introduction

The perception process on the human eye is based on the composition of multiple frequency components and sprectrums according to the visual distance, that is, frequency components vary inversely with respect to distance, which means that higher frequencies are to be seen at lower distances and lower frequencies are associated to higher distances, this implies that is possible to form an image which presents different low and high frequency components coming from two different distributions, in this case, images (Hybrid Image).

To form an hybrid image, it is possible to process the input images $(x_1, x_2)$ on the Fourier Frequency space and then extract their high and low frequency components by using Filters $(F_1, F_2)$ (Low-pass and High-pass, respectively) [1], after processing the images, it is possible to combine their responses in time to form a single image $(y)$, it is necessary to observe that both input images must have the same dimensions[1]. The process of hybrid image formation can be expressed as (1)

$$Y = I_1 \cdot H_1 + I_2 \cdot (1 - F_2) \tag{1}$$

## 2. Materials and Methods

To evaluate the feasibility of the procedure, an implementation was defined on Python, by using the OpenCV library to process basic Computer Vision operations, such as filter convolution and pyramid formation, Numpy was employed to manipulate image matrices and to apply operations in frequency domain, finally matplotlib was used to display and graph the results. To evaluate the synthesis method, different pair images were processed by using Gaussian Filters, designed in frequency domain and applied in time domain by using convolution, then to assess the final result a Gaussian Pyramid was built, it is expected that the high frequency components of the image can be seen on the first levels of the pyramid, but not in the last levels. With respect to the low frequency components, it is expected that the image can be seen on the last levels.

[1]To get a consistent result it is possible to modify and set the cutoff frequencies and the kernel sizes defined for both filters.

## 3. Results

After evaluating the method over different image pairs, it is possible to appreciate that the kernel size and the cutoff frequencies influence the result, for instance, if the cutoff frequency for the low pass filter is too high, the image may contain all the frequencies of the low-frequency image. Similarly, if the cutoff frequency for the high-pass filter is too low, then all the frequencies associated to the high-pass image may be reflected on the final result. With respect to the kernel size, it is possible to observe that increasing/decreasing the kernel size is equivalent to change the cutoff frequency. Additionally, to obtain consistent results, it is necessary that the images present similar geometry and pose orientations.

During the evaluation of the filters, it is possible to observe that applying the operations on the frequency space increases the memory consumption at the expense of the time complexity, especially if the images are large, for example to represent an image on frequency of dimension (6000, 4000, 3), it is necessary to use a space of dimension (6000, 4000, 2, 3), because the pixel intensities are transformed onto complex numbers. This implies that if the values are represented as floating numbers of 32 bits, then it is necessary to store 5.7Gb of data, compared to the computation time of a convolution in time with a lower memory footprint.

## 4. Conclusions

After evaluating the procedure to conform an Hybrid Image, it is possible to appreciate that different hyperparameters and geometry features of the input images can alter the final result, for instance, the cutoff frequencies and the kernel sizes of the filters implied on the process can atenuate or intensify the presence of the frequency components of the original images on the hybrid image. Also the procedure used to compute the final image may increase or decrease the memory footprint at the expense of the time complexity of the convolution operation, which means that the frequency/time tradeoff may be of interest when the dimensionality of the images change. Finally, synthetising hybrid images may be of interest on perception studies and art representations.

## Referencias

[1] Aude Oliva, Antonio Torralba y Philippe G. Schyns. “Hybrid Images”. En: *ACM Trans. Graph.* 25.3 (jul. de 2006), págs. 527-532. ISSN: 0730-0301. DOI: `10.1145/1141911.1141919`. URL: `http://doi.acm.org/10.1145/1141911.1141919`.

## 5. Code Snippets

```python
def pyramid_built_up(img, n):
    """
    Given an in imput image, build an image that contains n levels of the Gaussian Pyramid.

    Parameters
    ----------
    img: array_like
        Input image.
    n: int
        Number of pyramid levels to display.
    """
    composite = img
    last_lvl, cur_lvl = img, cv2.pyrDown(img)
    n -= 1
    for lvl in range(n):
       H, W, C = composite.shape
       mask = np.zeros((H, W + cur_lvl.shape[1], C))
       mask[-H:, 0:W] = composite
       mask[-cur_lvl.shape[0]:, W:W + cur_lvl.shape[1]] = cur_lvl
       composite = mask
       last_lvl, cur_lvl = cur_lvl, cv2.pyrDown(cur_lvl)
    return composite
```

*Listing 1:* Function which builds an image that contains $n$ levels of the Gaussian Pyramid associated to an input image

```python
lp1 = butterworth(size_1, lp_freq, order) # Low pass frequency response
lp1_t = np.abs(fft.ifft2(lp1)) # Time domain representation
lp2 = butterworth(size_2, hp_freq, order) # High pass frequency response
lp2_t = np.abs(fft.ifft2(lp2)) # Time domain representation
```

*Listing 2:* Declaration of high pass and low pass filters on frequency and time

```python
lp_img = cv2.GaussianBlur(img1, size_1, lp_freq) # Apply Gaussian filter (LP)
hp_img = img2 - cv2.GaussianBlur(img2, size_2, hp_freq) # Apply Gaussian filter (HP)

synth = lp_img + hp_img # Build Hybrid Image
```

*Listing 3:* Filter convolution and image addition to conform Hybrid Image

## Image results and Description

### Cold War

- **Description:** A mixture between Moscow Saint Basil Cathedral and New York City Statue of Liberty, these are good times for Humanity, isn't it?
- **Processing:** Original images were of size (6016, 4000) and (4000, 6016) pixels, respectively. The images were cropped and centered.
- **Parameters:** $\omega_{cl} = 0{,}1$, $\omega_{ch} = 21$, $|K_l| = 21 \times 21$, $|K_h| = 41 \times 41$
- **Execution invocation:**

```
python main.py  --lowpass 0.1 --highpass 21 --save output.jpg --lker_size 21 21
↪  --hker_size 41 41 ./data/Moscow.jpg ./data/Liberty.jpg
```

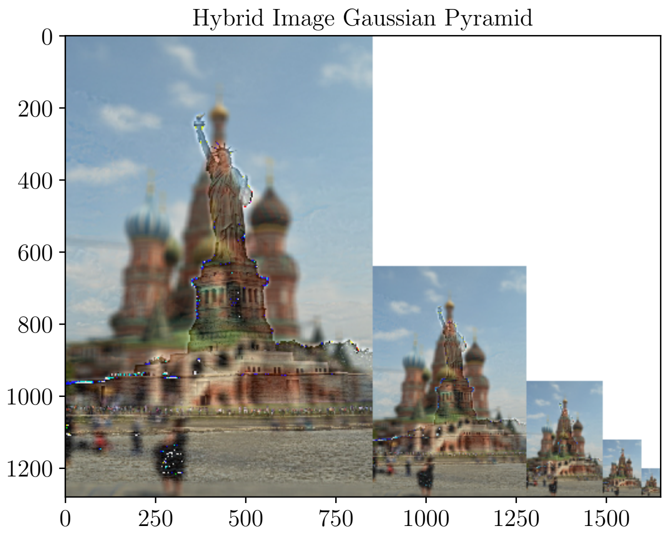

*Figura 1:* Gaussian Pyramid of level 5 *Cold War* image

### Two Face

- **Description:** Some friends decided to participate on this lab.
- **Processing:** One of the images was cropped and translated
- **Parameters:** $\omega_{cl} = 30$, $\omega_{ch} = 2$, $|K_l| = 11 \times 11$, $|K_h| = 3 \times 3$
- **Execution invocation:**

```
python main.py  --lowpass 30 --highpass 2 --save output.jpg --lker_size 11 11
↪  --hker_size 3 3 ./data/Face_2.jpg ./data/Face_1.jpg
```

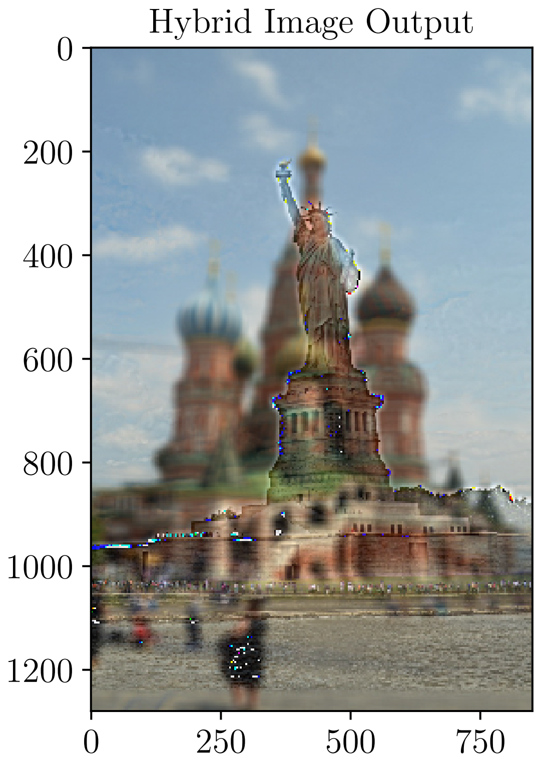

*Figura 2:* Hybrid image: *Cold War* image

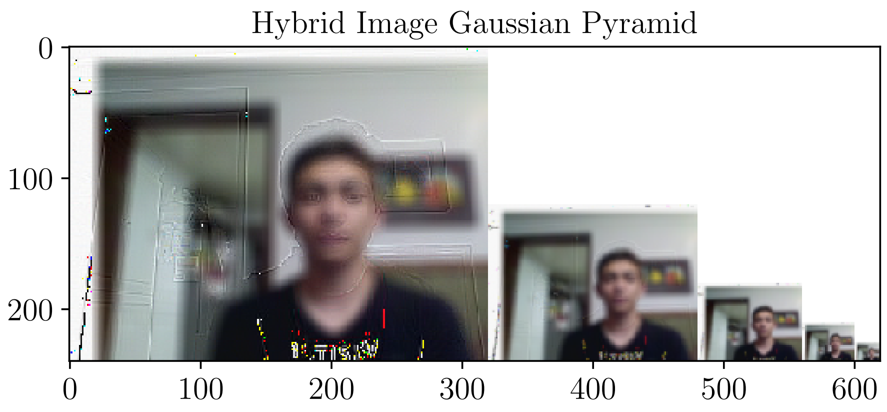

*Figura 3:* Gaussian Pyramid of level 5 *Two Face* image

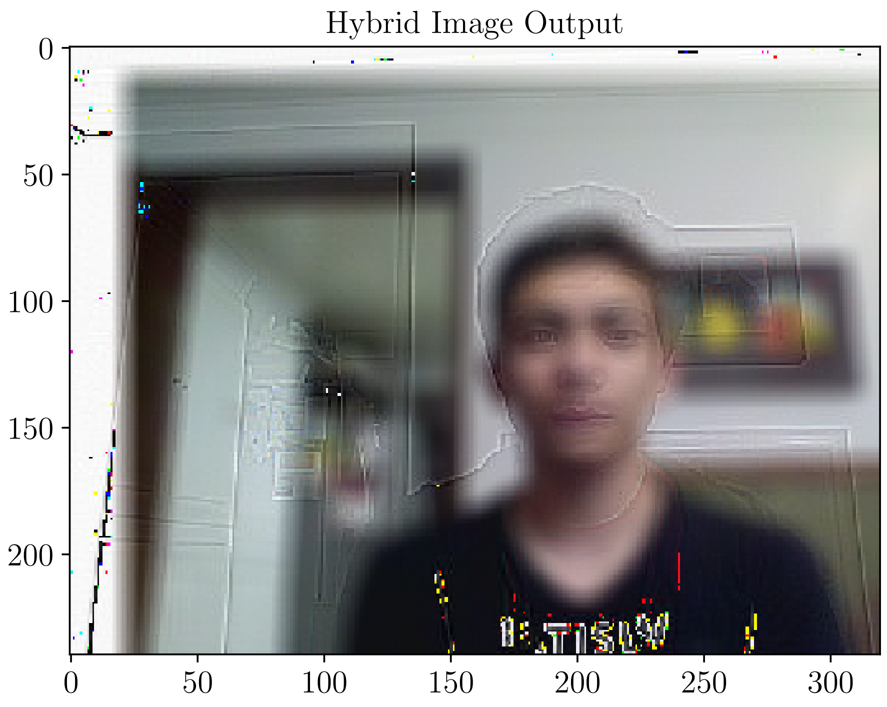

*Figura 4:* Hybrid image: *Two Face* image