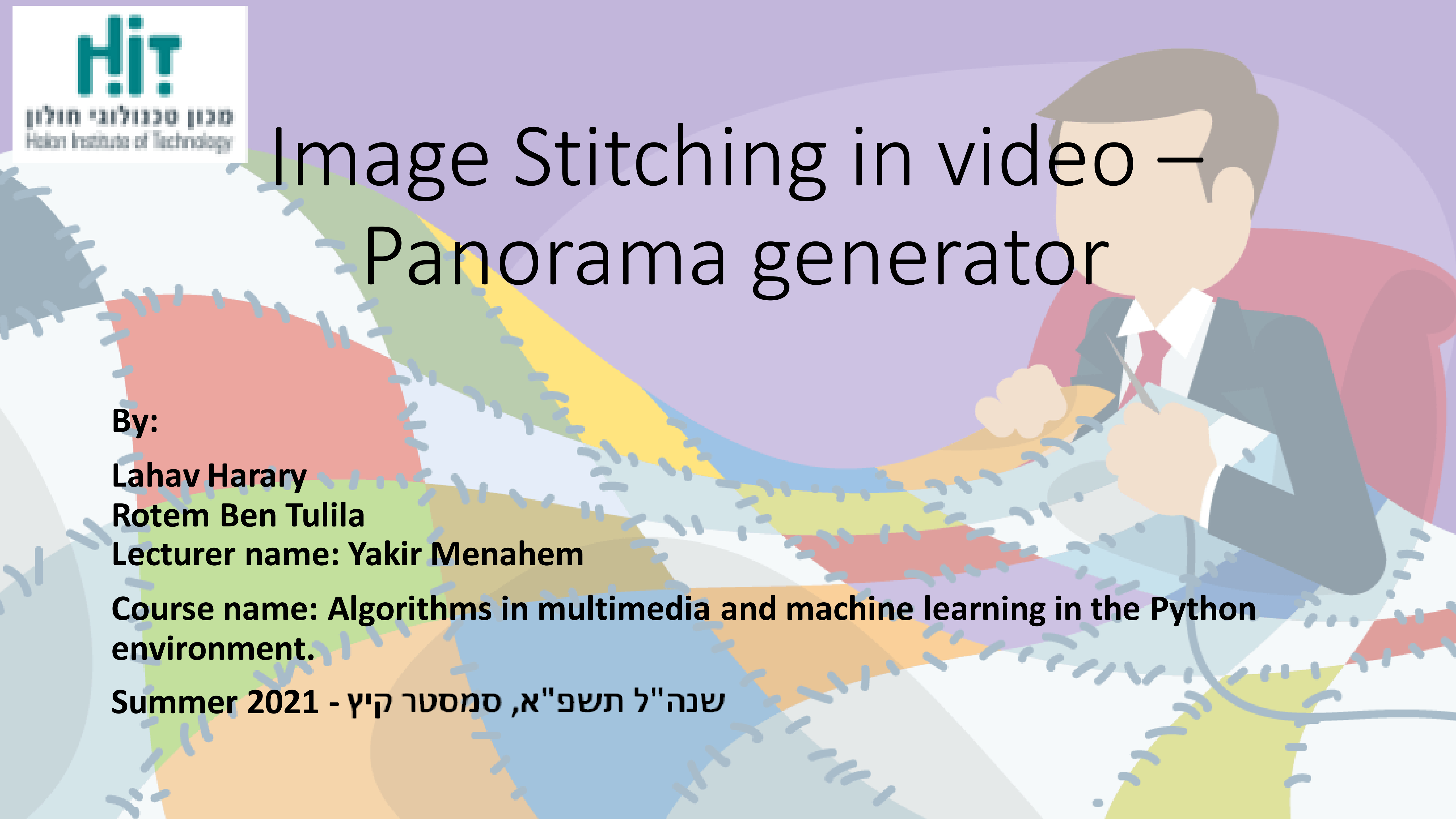


Image Stitching in video – Panorama generator

By:

Lahav Harary
Rotem Ben Tulila
Lecturer name: Yakir Menahem

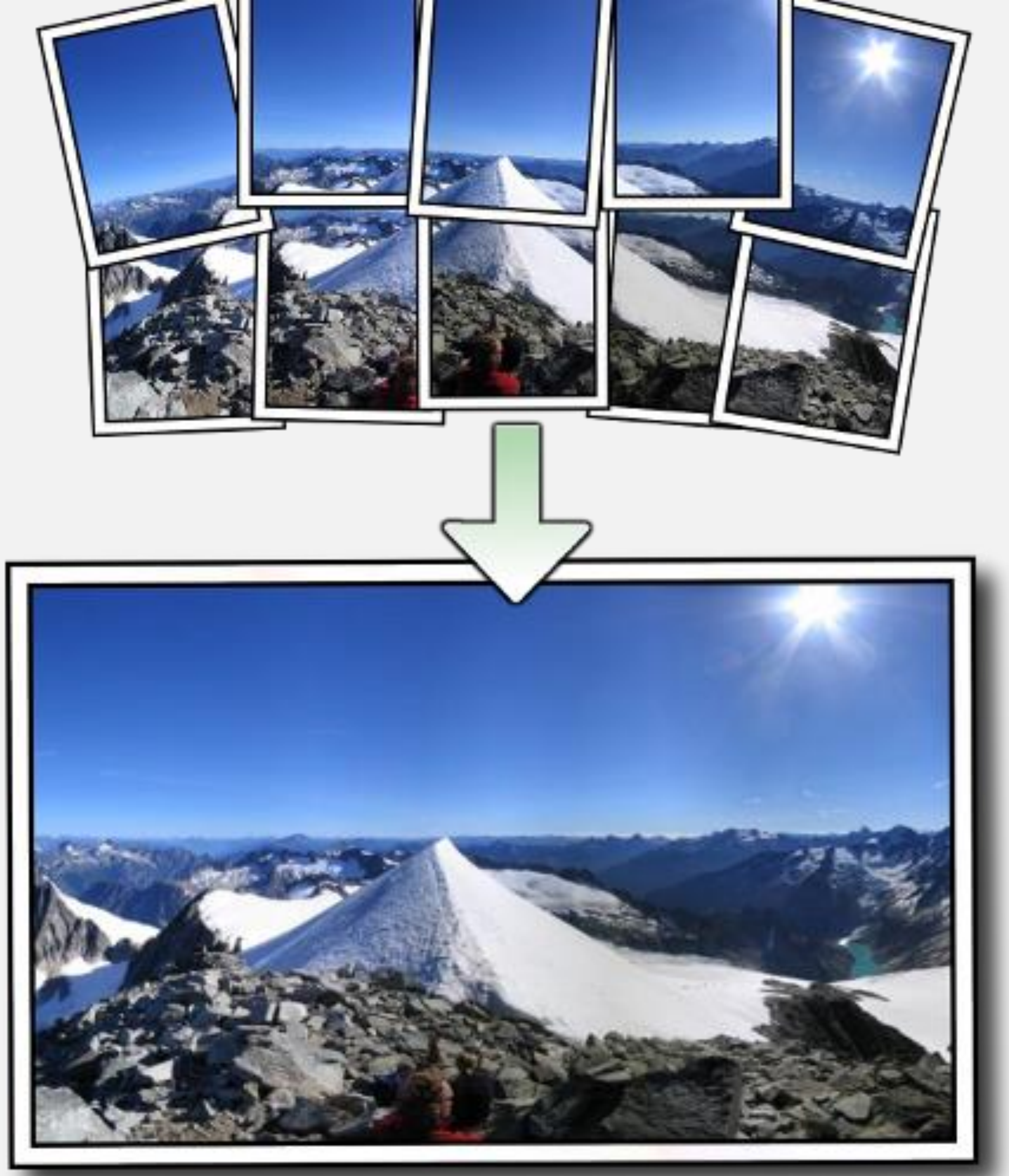
Course name: Algorithms in multimedia and machine learning in the Python environment.

Summer 2021 - שנה"ל תשפ"א, סמסטר קיץ

Project Background

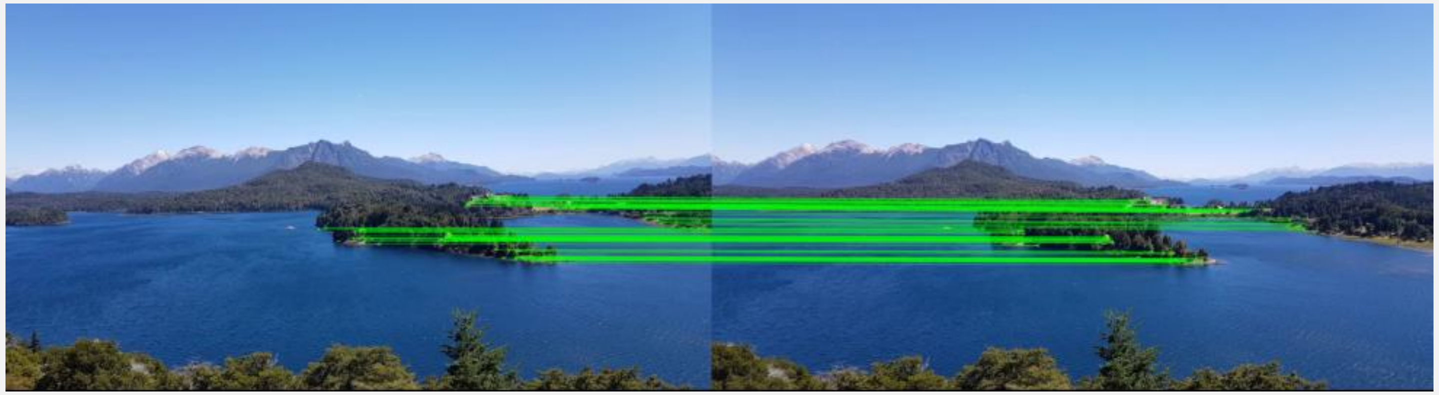
Creating a "panoramic image" A wide image that contains many objects. Sometimes it is necessary to take a wide picture, but a phone or camera is not enough.

Our project addresses this by turning a video into a panoramic image by stitching images.

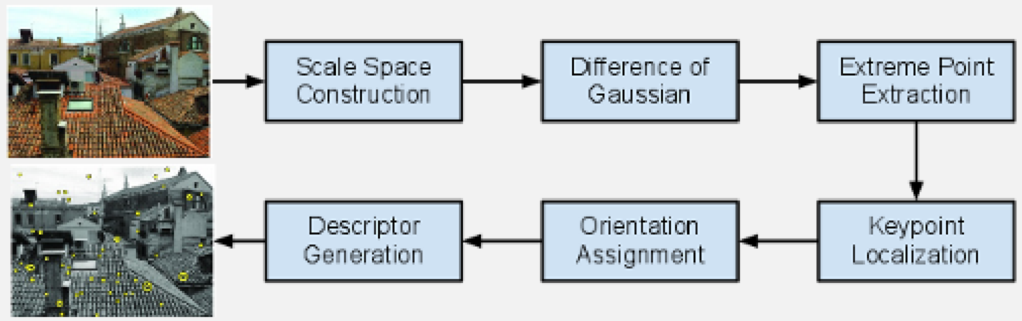


Different approaches for solving this problem

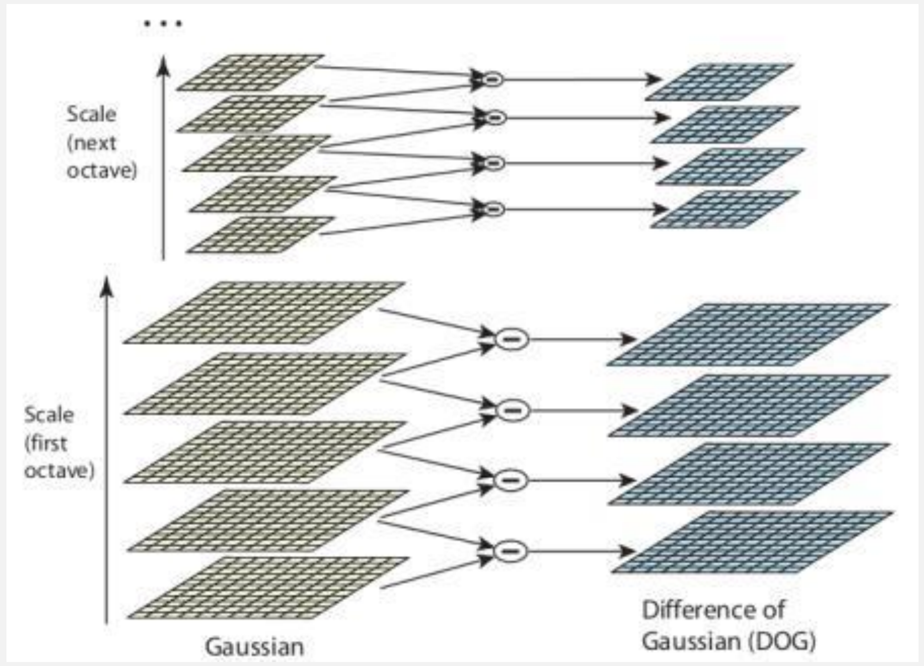
- The main problem in image stitching is finding the matching points between the images.
- In order to solve the problem, there are several main algorithms.
- We will talk a little bit about these algorithms help us solve the problem.
- The most used algorithms today are:
- 1. SIFT Algorithm – Scale Invariant Feature Transform.
- 2. SURF Algorithm – Speeded Up Robust Feature.
- 3. ORB Algorithm – Oriented FAST and Rotated BRIEF.



SIFT algorithm – Scale Invariant Feature Transformation:

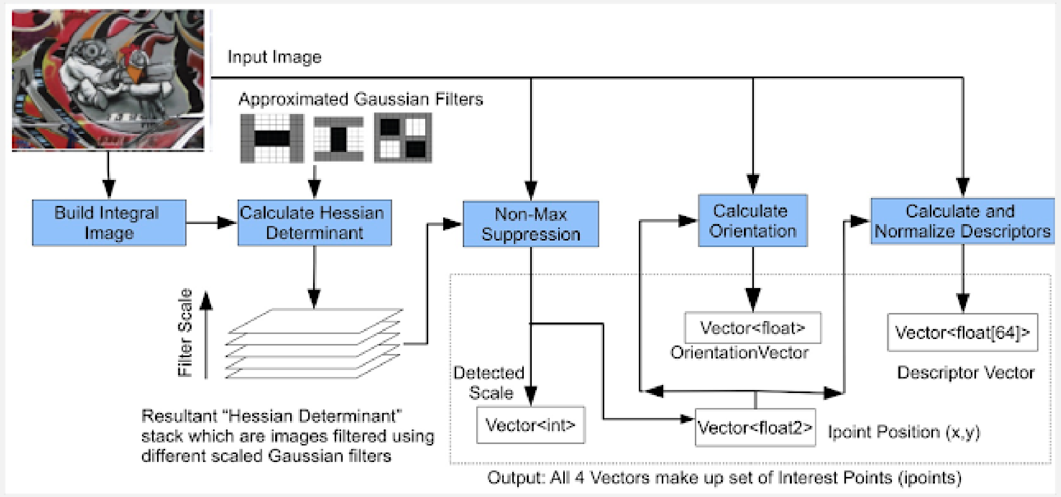


- In order to find the key points, we will use a method called "Difference of Gaussian" (DoG).
  This method takes the photo and uses gaussian blurring a couple of times with a different amount of blur for each one, later on it subtracts the images from each other. We will look on those photos as a Stack of images and we will look for extreme points – points that all of their neighbors on x and y axis are identical. After the stage is finished, we are left with key points.
- The next stage is computing our descriptors, we will do so by looking at the neighbors of the earlier found key points.
  We will take those neighbors and look at them as smaller pieces, later on we will compute the neighborhoods gradient.
- The gradients that were found earlier will be stored inside histograms and will be calculated in order to find our descriptors.



SURF algorithm: Speeded Up Robust Features

- SURF consists of three parts: Detection, Descriptor, Matching
- Detection: SURF uses a square-shaped filter as an approximation of Gaussian smoothing (called box filters). Interest points can be found of different scales, that's why box filters with difference sizes are applied on the photo. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space. Scale space interpolation is especially important in this case, as the difference in scale between the first layers of every octave is relatively large
- Descriptors: the goal of the descriptor is to provide a unique and robust description of an image feature. by describing the intensity distribution of the pixels within the neighborhood of the point of interest.
- Matching: By comparing the Descriptors from different images matching can be done.


Input Image
Approximated Gaussian Filters
Build Integral Image
Calculate Hessian Determinant
Non-Max Suppression
Calculate Orientation
Calculate and Normalize Descriptors
Filter Scale
Vector<float>
OrientationVector
Vector<float[64]>
Descriptor Vector
Detected Scale
Vector<int>
Vector<float2>
Ipoint Position (x,y)
Resultant "Hessian Determinant" stack which are images filtered using different scaled Gaussian filters
Output: All 4 Vectors make up set of Interest Points (ipoints)

Introduction to ORB algorithm: Oriented FAST and Rotated BRIEF

ORB is a feature detection algorithm just like SIFT and SURF.
ORB preforms as well as SIFT (which is better than SURF) but uses almost half of the time.

ORB builds FAST key point detector and the BRIEF descriptor with good performance and low cost.

ORB’s main contributions are as follows:

- The addition of a fast and accurate orientation component to FAST.
- The efficient computation of oriented BRIEF features.
- A learning method for decorrelating BRIEF features under rotational invariance.

The main algorithm that we chose to work with:

- Why we chose to work with ORB:

  ORB performs as well as SIFT on the task of feature detection (and is better than SURF) while being a lot faster. ORB builds on the well-known FAST key point detector and the BRIEF descriptor. Both of these techniques are attractive because of their good performance and low cost

Advantages:

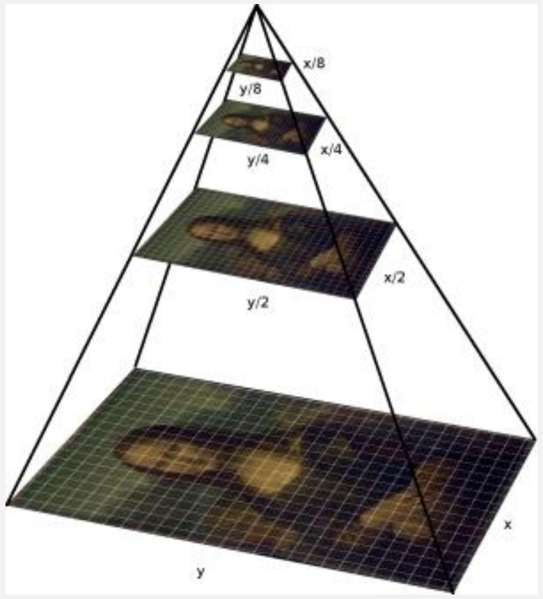
- ORB is faster than SIFT and SURF.
- ORB : 256 bit and SIFT: 4096 bit.
- ORB supply binary descriptors.

Disadvantages:

- ORB use two separate algorithms (FAST and BRIEF).
- ORB is not as robust to scale changes as SURF

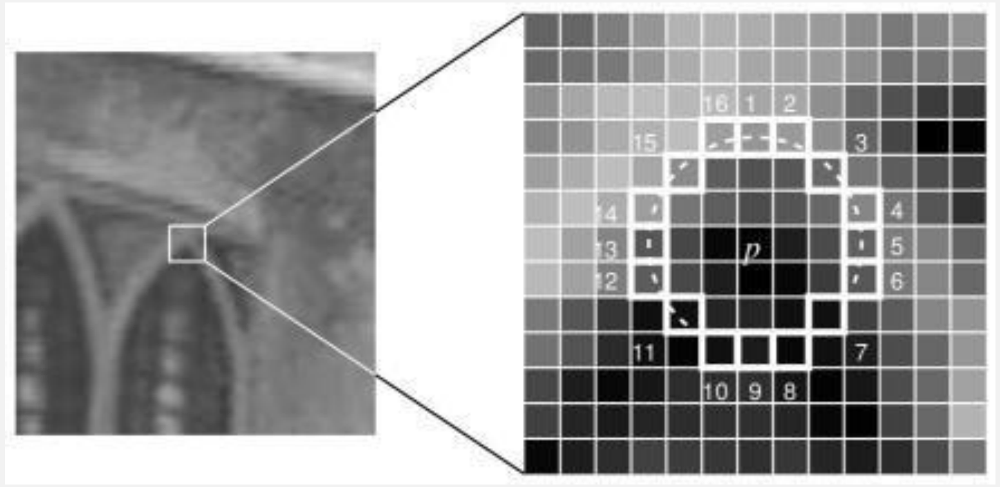
ORB algorithm : FAST

- FAST features do not have an orientation component and multiscale features.
- ORB uses a multiscale image pyramid that multiscale representation of a single image, that consist of sequences of images all of which are versions of the image at different resolutions.
- Each level in the pyramid contains the downsampled version of the image than the previous level.
- Once orb has created a pyramid it uses the FAST algorithm to detect key points in the image. By detecting key points at each level orb is effectively locating key points at a different scale. In this way, ORB is partial scale invariant.



ORB algorithm : FAST

- Take a pixel p in an array FAST compares the brightness of p to surrounding 16 pixels that are in a small circle around p.
- Pixels in the circle is then sorted into three classes (lighter than p, darker than p or similar to p).
- If more than 8 pixels are darker or brighter than p than it is selected as a key point. So key points found by FAST gives us information of the location of determining edges in an image.

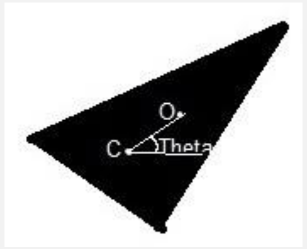


ORB algorithm : FAST

- After locating key points orb now assign an orientation to each key point like left or right facing depending on how the leve ls of intensity change around that key point. For detecting intensity change orb uses intensity centroid.
- The intensity centroid assumes that a corner's intensity is offset from its center, and this vector may be used to impute an orientation.
- First, the moments of a patch are defined as: $$m_{pq} = \sum_{x,y} x^p y^q I(x,y)$$
- With these moments we can find the centroid, the "center of mass" of the patch as: $$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$$
- We can construct a vector from the corner's center O to the centroid -OC. The orientation of the patch is then given by:

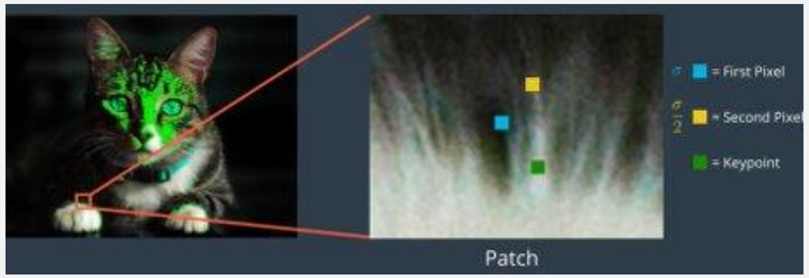
$$\theta = \text{atan2}(m_{01}, m_{10})$$

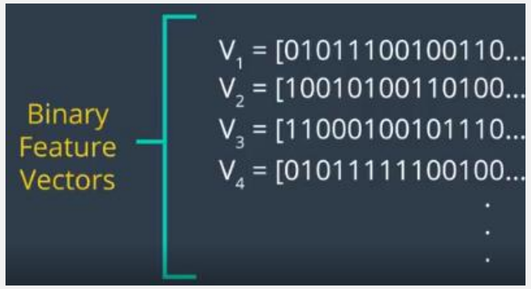
- Once we've calculated the orientation of the patch, we can rotate it to a canonical rotation and then compute the descriptor, thus obtaining some rotation invariance.



ORB algorithm : BRIEF

- Brief takes all key points found by the FAST algorithm and convert it into a binary feature vector so that together they can represent an object.
- Binary features vector also known as binary feature descriptor is a feature vector that only contains 1 and 0. In brief, each key point is described by a feature vector which is 128–512 bits string.
- Brief start by smoothing image using a Gaussian kernel in order to prevent the descriptor from being sensitive to high-frequency noise. Than brief select a random pair of pixels in a defined neighborhood around that key point.
- The first pixel in the random pair is drawn from a Gaussian distribution centered around the key point with a stranded deviation or spread of sigma. The second pixel in the random pair is drawn from a Gaussian distribution centered around the first pixel with a standard deviation or spread of sigma by two.
- Now if the first pixel is brighter than the second, it assigns the value of 1 to corresponding bit else 0




Binary Feature Vectors
V_1 = [01011100100110...
V_2 = [10010100110100...
V_3 = [11000100101110...
V_4 = [01011111100100...
⋮

ORB algorithm : BRIEF

- Again, brief select a random pair and assign the value to them. For a 128-bit vector, brief repeat this process for 128 times for a key point. Brief create a vector like this for each key point in an image. However, BRIEF also isn’t invariant to rotation, so orb uses rBRIEF(Rotation-aware BRIEF).
- Consider a smoothed image patch, p. A binary test τ is defined by:

$$\text{Where } \tau(p; x, y) \text{ is defined as :}$$
$$\tau(p; x, y) = \begin{cases} 1 & : p(x) < p(y) \\ 0 & : p(x) \geq p(y) \end{cases}$$
$$p(x) \text{ is the int ensity value at pixel } x.$$

- where p(x) is the intensity of p at a point x. The feature is defined as a vector of n binary tests:

$$f(n) = \sum_{1<i<n} 2^{i-1} \tau(p; x_i, y_i)$$

- The matching performance of BRIEF falls off sharply for in-plane rotation of more than a few degrees. ORB proposes a method to steer BRIEF according to the orientation of the key points. For any feature set of n binary tests at location (xi, yi), we need the 2 x n matrix:

$$S = \begin{pmatrix} x1, \ldots x_n \\ y1, \ldots y_n \end{pmatrix}$$

ORB algorithm : BRIEF

- It uses the patch orientation θ and the corresponding rotation matrix Rθ, and construct a steered version Sθ of S:

$$S_\theta = R_\theta S$$

- Now, the steered BRIEF operator becomes:

$$g_n(p,\theta) = f_n(p) \mid (\mathbf{x}_i, \mathbf{y}_i) \in S_\theta$$

- Than it discretizes the angle to increments of 2π/30 (12 degrees), and construct a lookup table of precomputed BRIEF patterns. As long as the key point orientation θ is consistent across views, the correct set of points Sθ will be used to compute its descriptor.

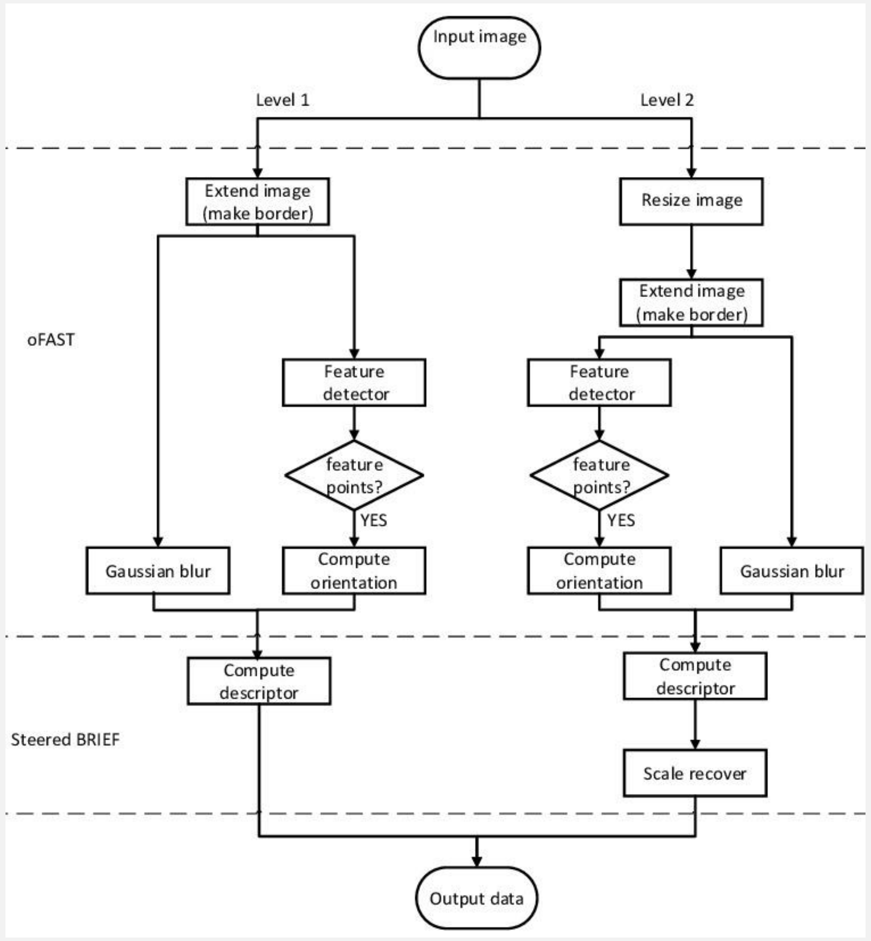
$$S = \begin{pmatrix} x1, \ldots x_n \\ y1, \ldots y_n \end{pmatrix}$$

ORB algorithm : BRIEF

ORB specifies the rBRIEF algorithm as follows:

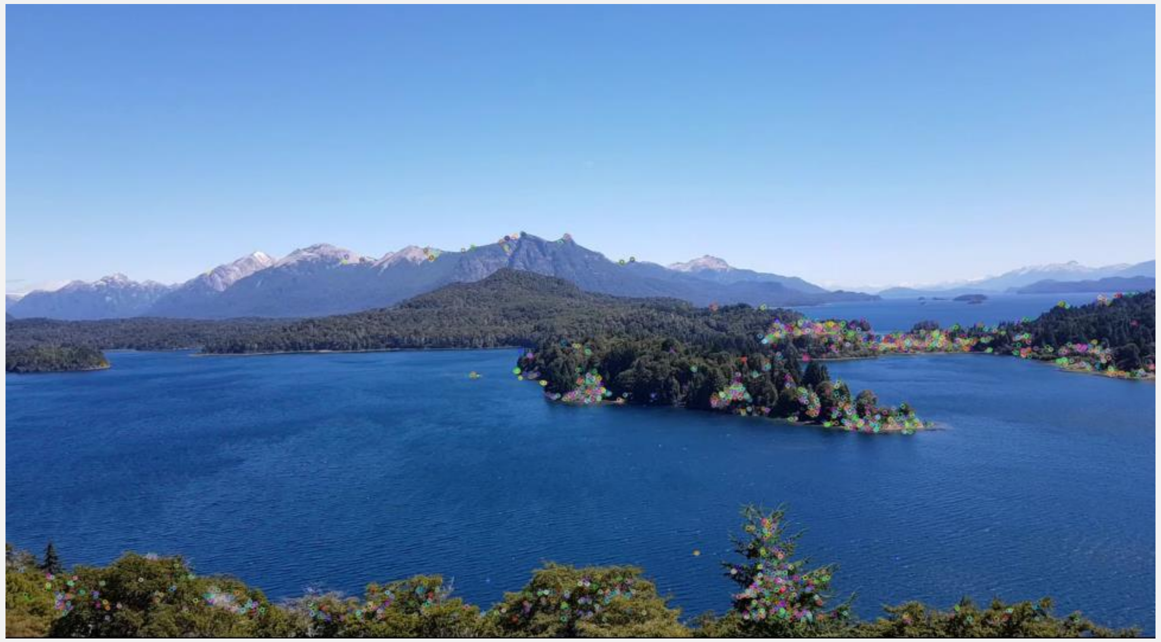
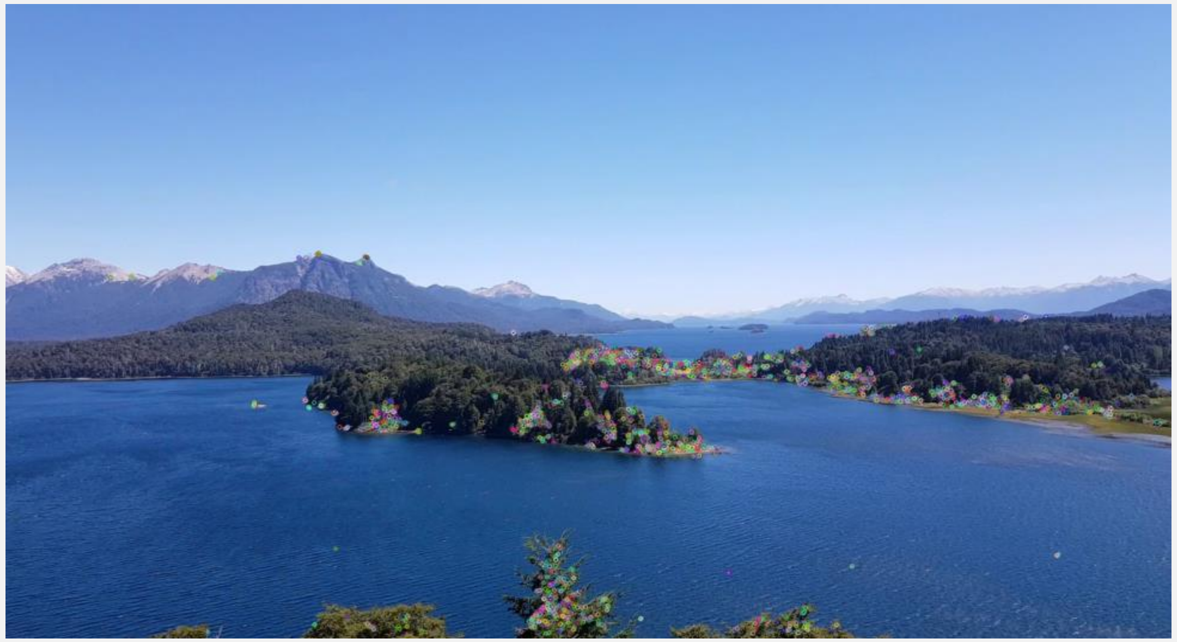
- Run each test against all training patches.
- Order the tests by their distance from a mean of 0.5, forming the vector T.
- Greedy search:
    - Put the first test into the result vector R and remove it from T.
    - Take the next test from T, and compare it against all tests in R. If its absolute correlation is greater than a threshold, discard it; else add it to R.
    - Repeat the previous step until there are 256 tests in R. If there are fewer than 256, raise the threshold and try again
    - rBRIEF shows significant improvement in the variance and correlation over steered BRIEF.

ORB algorithm: Flow



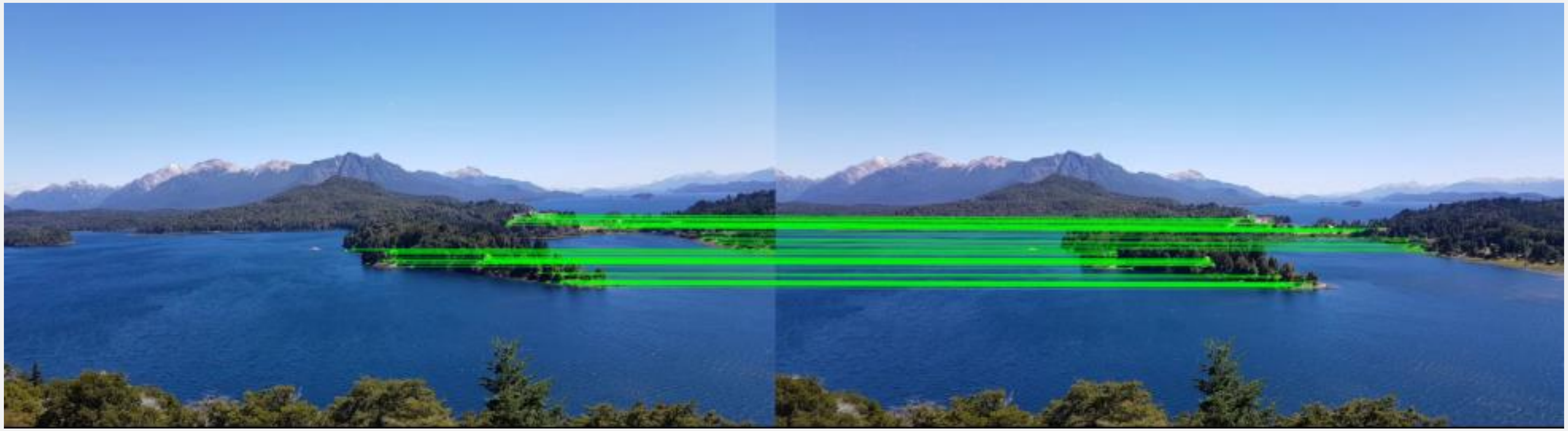
Final product demonstration:

Key points:

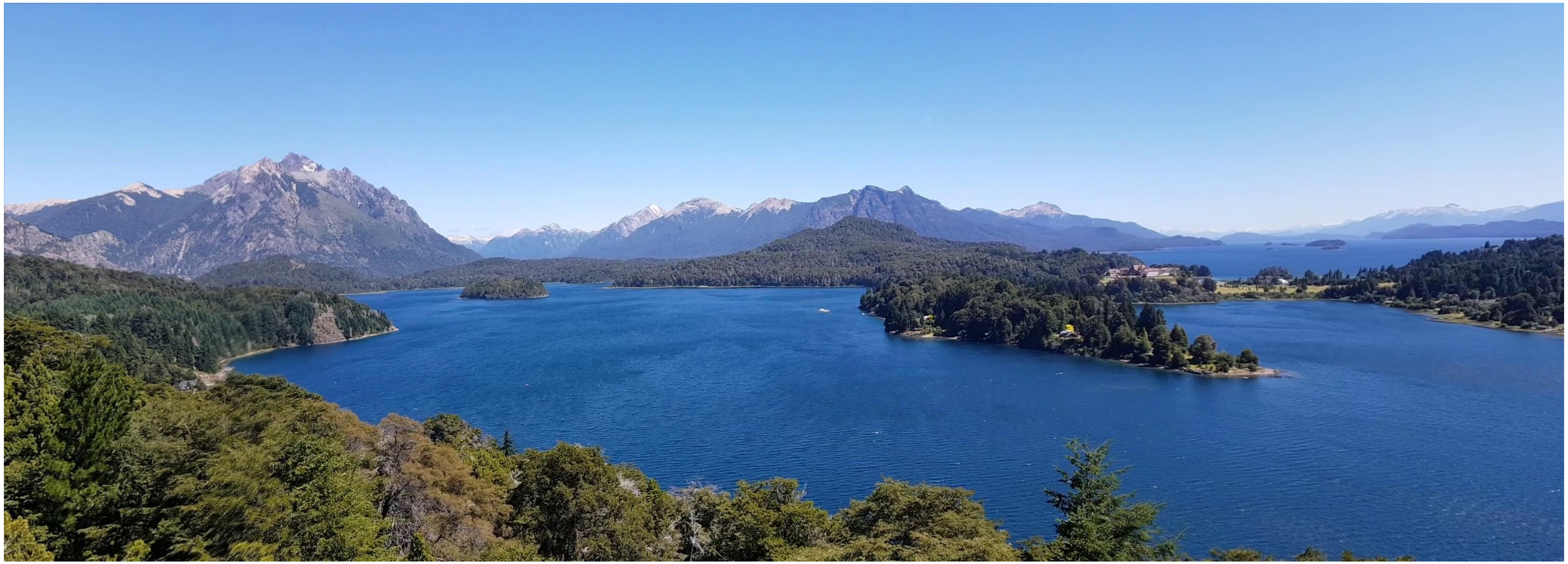
Final product demonstration:

- Matching the key points:



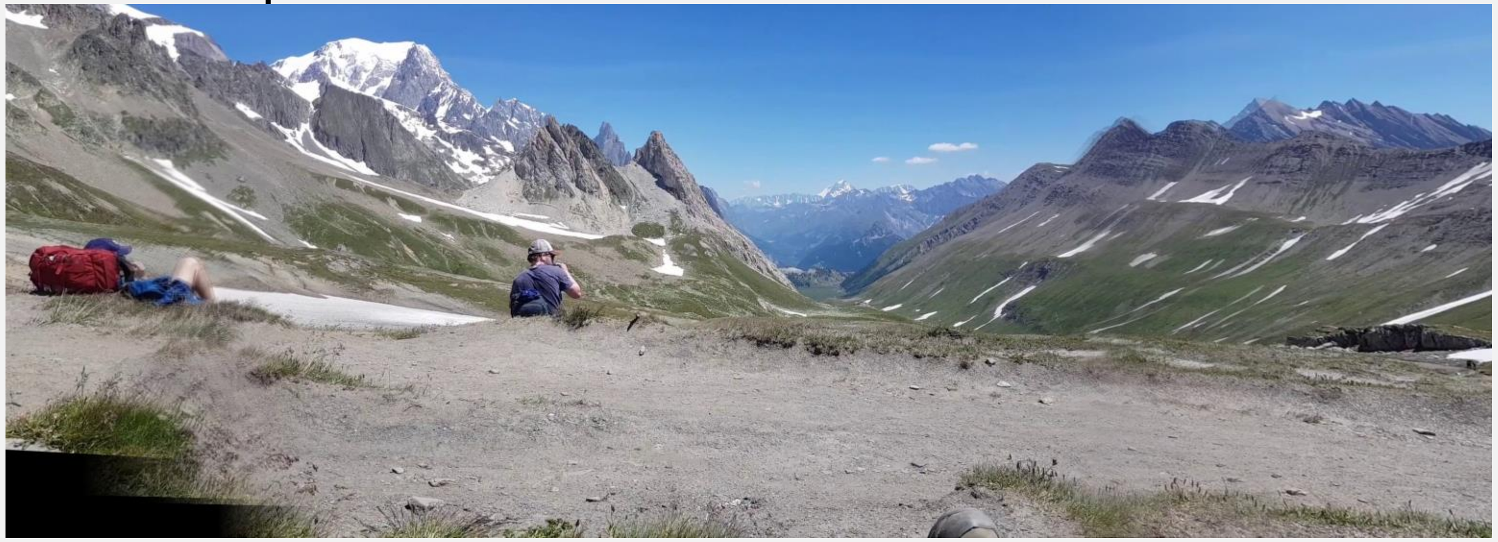
Final product demonstration:

- Result:



Final product demonstration:

- Not all videos work great, Unfortunately our algorithm doesn't do miracles
- If a video has too much tilt (Up or Down) we can't make a good panorama out of it. Here is an example of that:



- Notice the black mark at the bottom left side of the picture.

Link to the project video:

- The link: Promotial Video- Panorama Generator