The pipeline to detect AR Tags is divided in the next steps:

- Gaussian filter: First we smooth the image by applying a Gaussian filter to it. This removes noise to make the detection of the tag easier and avoids false detections.

- The next step is to detect the edges in the image. This is done by using the `edge()` MATLAB function. We tried different methods and finally chose Roberts method because of its result. Since the output of this function is a binary image, we apply morphological operations to it. First we apply erosion to remove small and noisy blobs. Once this is done, dilation is applied to make the desired edges more consistent.

- Once the edges have been detected, we get the boundaries created by these edges. The used function is `bwboundaries()`. This function returns the positions of all the points that create a boundary in a vector. It also returns parent boundaries with their inner boundaries. This is useful since the markers boundaries are inside the white papers boundaries.

- With all the boundaries points already known, the next step is to apply an algorithm to detect the corners of all the quadrilaterals in the image. After trying different methods as the Harris corner detector, we decided to implement the Douglas-Peucker algorithm. This algorithm takes a boundary vector (or multiple connected points) as an input and simplifies it to the minimum polygon by using line simplification. It returns the list of corners in the boundary. To implement the Douglas-Peucker algorithm, we used the code "Line Simplification" by Wolfgang Schwanghart found in Mathworks File Exchange. The link to it can be found at the beginning of our code. The function is included in the scripts folder and it's called `dpsimplify()`.

- To detect the quadrilaterals, just after the Douglas-Peucker algorithm has been used in each of the boundaries, we apply a condition in which every simplified polygon with 4 corners is going to be a quadrilateral.

- To detect multiple quadrilaterals at the same time, we basically run the previous step for each parent and inner boundaries in the image. The next steps are also done for each of the quadrilaterals detected in the image.

- To remove some of the false quad detections, an area filter is applied.

- The next step is to transform the world projection of the detected quadrilaterals into the image plane. This is done by using the four detected corners and the four corners given in the reference marker image to obtain the homography matrix that relates them. The homography is calculated with the MATLAB function `fitgeotrans()` or Peter Kovesi's function `homography2d()`. To get the unwarped representation of the projected marker, the function `imwarp()` is used.

- Once we have obtained the transformed image post the homography , we design the grid to encode the marker and detect its id number. Since the size of the transformed image is known, we encode a 8x8 grid on to it and use the format of the grid to eliminate any false detections (i.e. there should exist a padding of 0s as a margin for the image. We also use the fact that one of the corners of the inner cube must be white to get the orientation of the marker).

- Once we have obtained the grid and the orientation of the marker, it is a simple process to code the inner 2x2 tag to detect the ID based on the significant bits.