CEC495A
Lecture 11
RANSAC Video Stabilization

Akhan Almagambétov
Akhan.Almagambetov@ERAU.edu
Embry-Riddle Aeronautical University
Prescott, Arizona

I. Background

In this lecture, we will be discussing the theory behind RANdom SAmple Consensus (RANSAC), which can be used for image stitching (i.e. making panoramas out of discrete images) as well as video stabilization (which has valuable applications in ground and air surveillance). Using Harris corner detection and RANSAC, the most likely image transformation will be determined and two sequential images will be "stabilized". We will test the algorithm with dashboard camera footage when driving on rough terrain, as well as an aerial far-infrared surveillance video (similar to those captured by Unmanned Aerial Vehicles (UAV)).

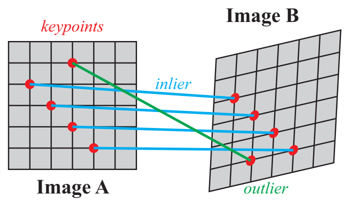
II. GitHub Repository

Retrieve the required lecture files from the GitHub repo. The folder you will need is **/ransac**. In this assignment, we will be working with image sequences extracted from shaky videos.

cec495a / lectures / lecture11

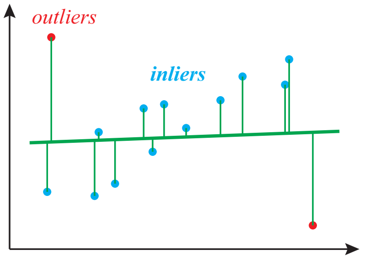
A. RANSAC

Consider a typical scenario, where you are required to match two similar images (i.e. sequential images from a "shaky" video or two images that need to be stitched into a panorama).



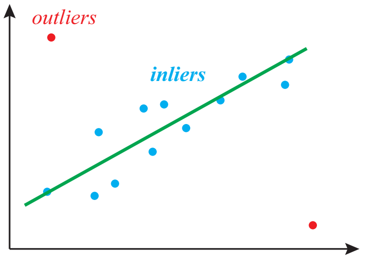
A good way to do this would be to generate keypoints in both images and somehow match them (either using correlation or, just like we did with SURF keypoints, match them using the descriptor distances). Regardless of which approach you take, there will exist *outliers*—bad data points that do not fit a particular transformation model (and might even result in an incorrect model if they are used). Trying to eliminate these outliers so that they do not affect the transformation model is a challenge solved by the RANdom SAmple Consensus (RANSAC) algorithm, developed in 1981 by Martin Fischler and Robert Bolles.

Let's look at a simpler example, in which we try to fit a line to the points present in the 2D point space.

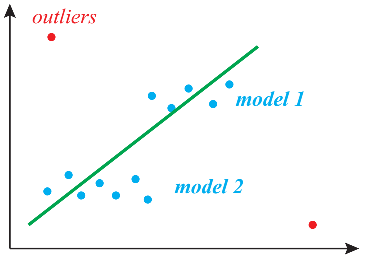


If we are to use the least-squares regression that we commonly use for fitting data points, the outliers are considered valid data, which results in an incorrect fit.

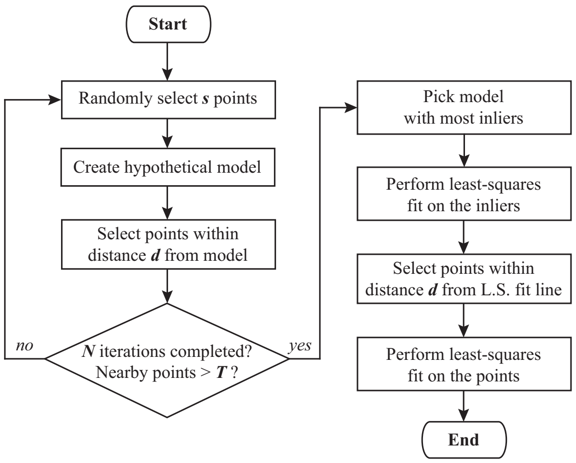
RANSAC is a robust estimation method that can detect and "overlook" outliers, making a more accurate fit.



As good as it is, RANSAC does not deal well with multiple models being present in the same point set (although this isn't much of a problem with video stabilization).



Here's a simplified flowchart for the RANSAC algorithm:

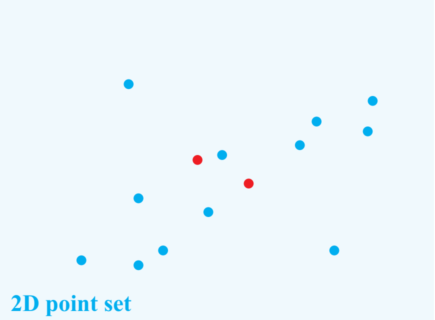

Start
Randomly select *s* points
Create hypothetical model
Select points within distance *d* from model
N iterations completed? Nearby points > *T* ?
no
yes
Pick model with most inliers
Perform least-squares fit on the inliers
Select points within distance *d* from L.S. fit line
Perform least-squares fit on the points
End

There are two important (and potentially detrimental) assumptions made in the RANSAC algorithm pertaining to the samples, which you must be aware of when using it:

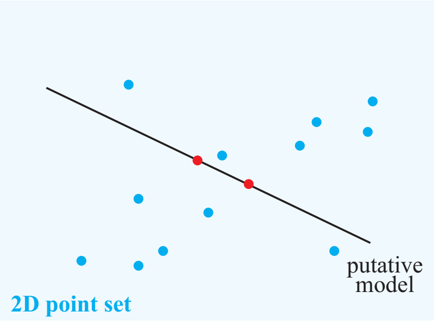
1) the outliers won't *consistently* skew the results in favor of an incorrect model, and
2) there exist *enough* "good" points to generate a consensus on a correct model.

Let's look at how RANSAC works, given a sample space of points with outliers.

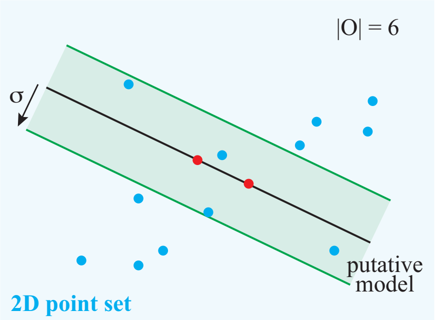
First, a random selection of s points is made, where s is the least number of points required to represent a model (in our case, we only need two points to accurately represent the model—given in **red** in the figure below).



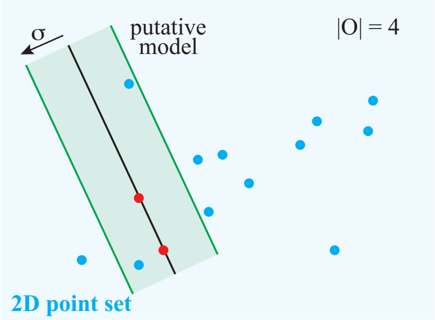
A hypothetical (putative) model is created based on s points:

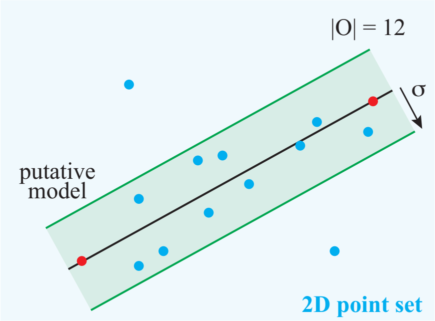


After this all points that are within distance d (or within one standard deviation σ–given by the user) of the model are counted. In our case, the number of points is $|O| = 6$.

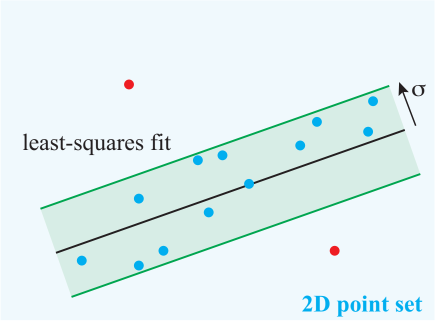


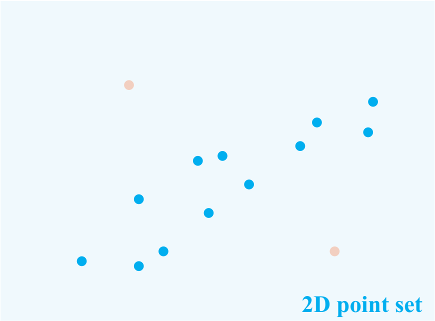
This process is repeated by picking random sets of s points until N iterations have been reached or the number of nearby points exceeds $T = (1 - e) * n$, where n is the total number of points in the 2D data set and e is the probability that the point is an outlier (so $(1 - e)$ is the probability that a point is an *inlier*).





Since $|O| = 12$ in the last figure, after N iterations are completed (or an iteration with the number of inliers $> T$ is found), this model is picked, as it contains the greatest number of points. The inliers are then fitted using the least-squares regression, after which the inliers around the L.S. line are considered. All of the outliers are discarded.





The number of iterations N are picked by solving for N in the following formula, where e is the probability that a point is an outlier, s is the least number of points with which a model can be accurately represented, and p is the desired probability of obtaining a valid sample.

$$1 - \overbrace{[\underbrace{1 - \overbrace{(\underbrace{1-e}_{\text{probability of an inlier}})^{s}}^{\text{prob. that sample contains only inliers}}}_{\text{prob. that 1+ sample points is an outlier}}]^{N}}^{\text{probability that N samples are outliers}} = p$$

Solving for N, we get:

$$N = \frac{\log(1-p)}{\log[1-(1-e)^s]} \tag{1}$$

To sum up RANSAC, the good points of the algorithm are that:

- it is robust to outliers, and
- repeatable results can be obtained with a sufficiently large number of iterations N.

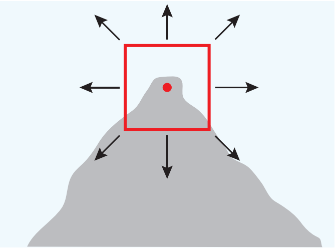
The drawbacks for RANSAC are that:

- it is very computationally expensive: the computational time grows quickly with the number of outliers, and
- the algorithm is not very good for categorizing multiple clusters of inliers (as seen previously).

B. Harris Corner Detector

In this demonstration, we will be using the Harris corner detector (1988 by Harris and Stephen). Although it is not recommended for real-time image processing applications due to its computational expensiveness, it's one of the simplest approaches to keypoint generation. In your next assignment, you will be using SURF to generate keypoints that will be integrated with the RANSAC-based video stabilization.

In a Harris detector, a window of width w is centered at a pixel in an image and then shifted in different directions to determine whether or not it is positioned on a corner (you can see where the computational complexity comes into play).



III. Video Stabilization

The video stabilization algorithm that we will be looking at today has five steps:

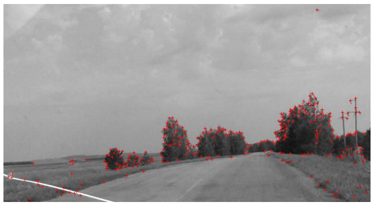
1) **Keypoint Generation**: feature points are generated in the two subsequent frames that we will be working with (in our case, using a Harris corner detector).
2) **Keypoint Matching**: the detected feature points are matched (using correlation or based on the descriptor vector distances, if using SURF).
3) **RANSAC**: matches consistent with an affine image transformation are found, others are discarded.
4) **Homography**: the image transformation is computed based on the matched keypoint pairs.
5) **Registration** (alignment): the image is transformed based on the Homography matrix calculated in the previous step.

A. Keypoint Generation

The keypoints are generated using the Harris corner detector (using the **harris** function). **HarrisThresh** determines how many keypoints are generated: if you lower the threshold, more points are generated, and you can eliminate keypoints by increasing the threshold. Note that in order for image registration to work properly, you need a large number of keypoints. The Harris corner detector required grayscale images as input, which you can convert using the **rgb2gray** function.

The **cim1** and **cim2** variables contain the corner images, but we will not be using them.

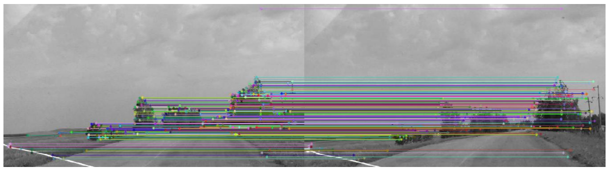
```
HarrisThresh = 10;

[cim1, r1, c1] = ...
   harris(Im1, 1, HarrisThresh, 3);
[cim2, r2, c2] = ...
   harris(Im2, 1, HarrisThresh, 3);
```

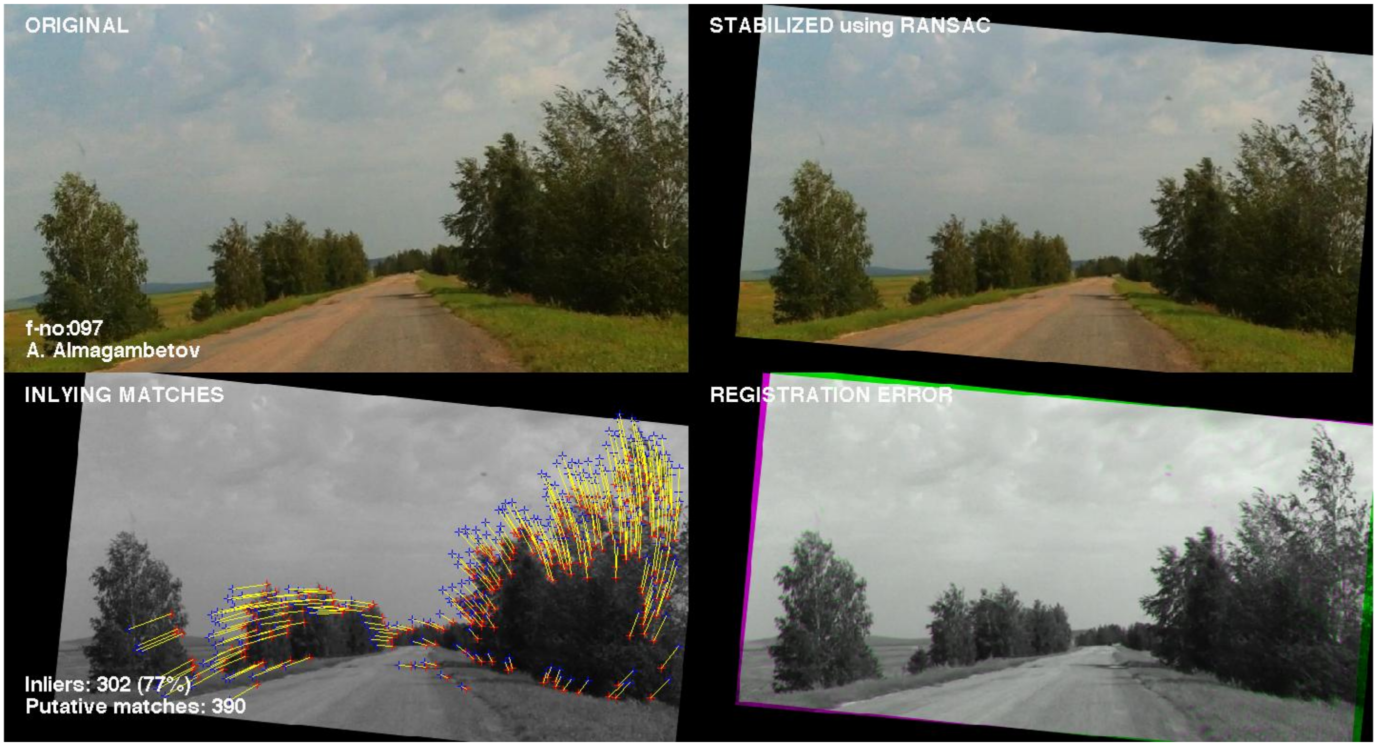


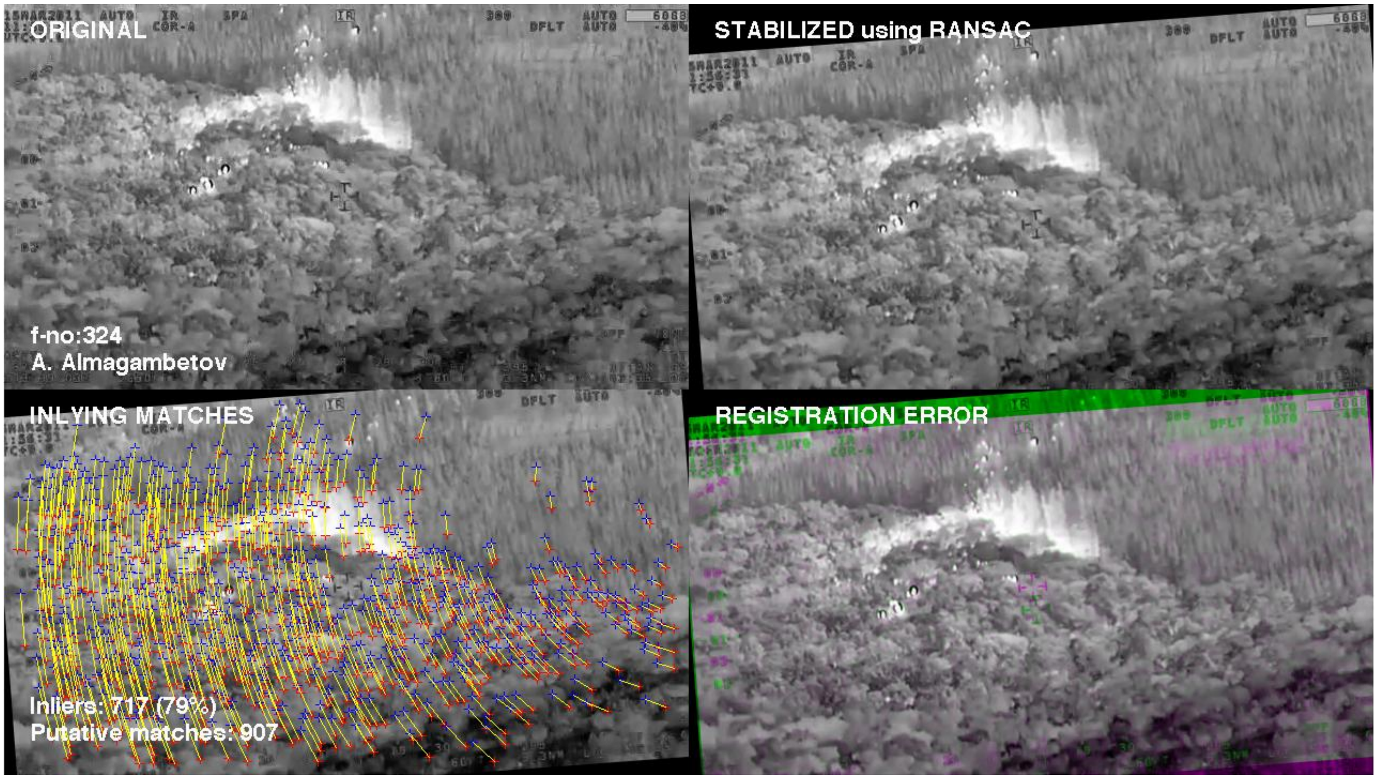
B. Keypoint Matching

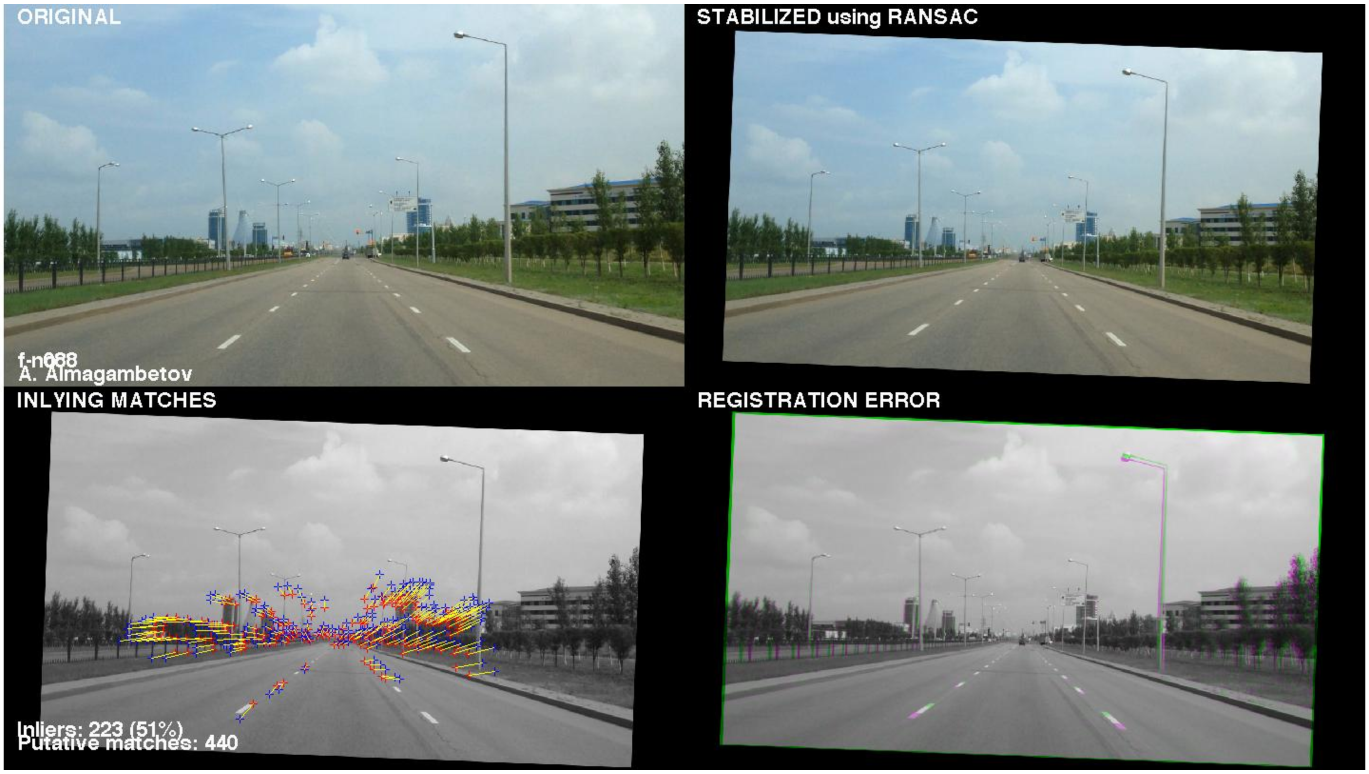
Keypoints are roughly matched using correlation. Although the matches are not perfect, this method allows us to match keypoints from the two images quickly and relatively inexpensively, in terms of computational load).

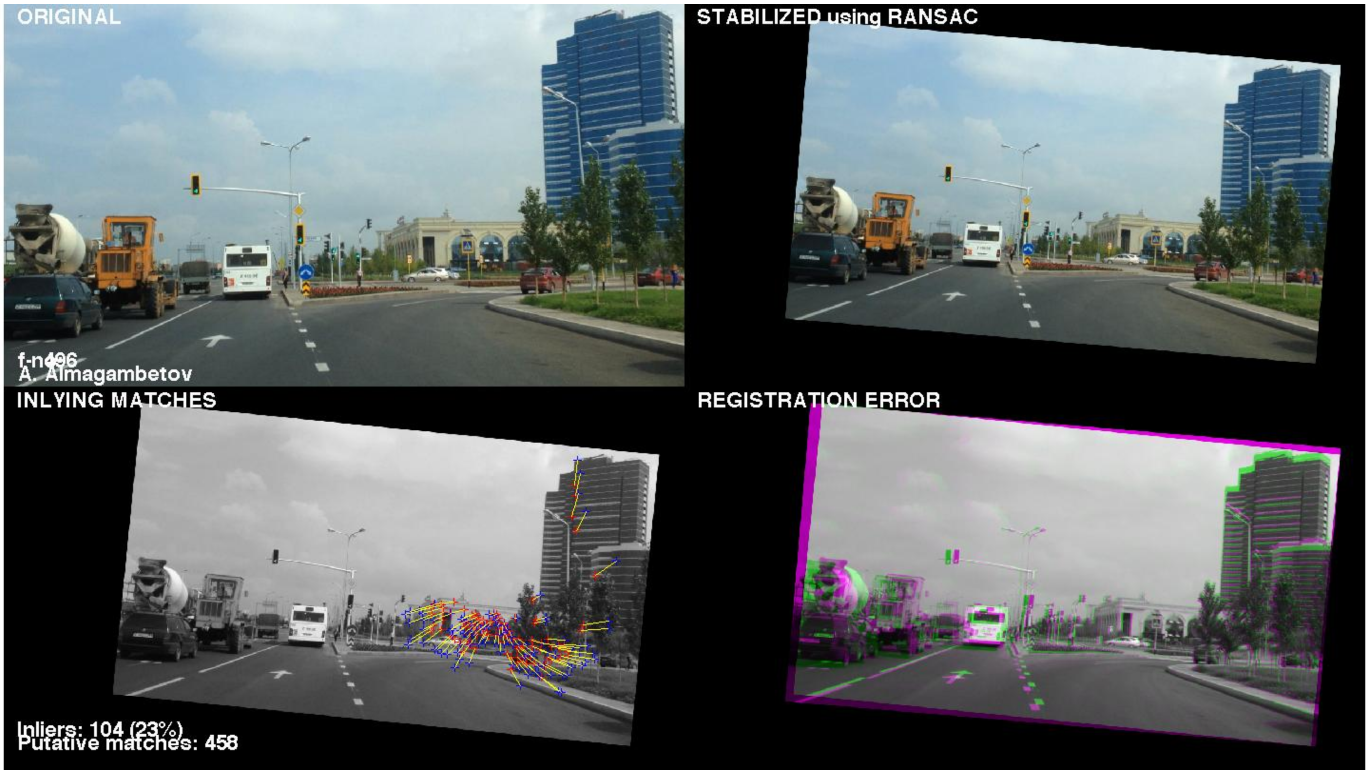
```
[m1,m2] = matchbycorrelation( ...
   Im1, [r1';c1'], Im2, [r2';c2'], 21, 50);
```




ORIGINAL
STABILIZED using RANSAC
f-no:097
A. Almagambetov
INLYING MATCHES
REGISTRATION ERROR
Inliers: 302 (77%)
Putative matches: 390


ORIGINAL
STABILIZED using RANSAC
f-no:324
A. Almagambetov
INLYING MATCHES
REGISTRATION ERROR
Inliers: 717 (79%)
Putative matches: 907


ORIGINAL
f-n088
A. Almagambetov
STABILIZED using RANSAC
INLYING MATCHES
Inliers: 223 (51%)
Putative matches: 440
REGISTRATION ERROR


ORIGINAL
f-n496
A. Almagambetov
STABILIZED using RANSAC
INLYING MATCHES
Inliers: 104 (23%)
Putative matches: 458
REGISTRATION ERROR

■