Smart Shot

Finding corresponding points on two cameras watching the same view from different positions

Boyko Konstantin
Rechister Lev

1) **Requirements**

2) **Overview**

3) **Design**

4) **Testing results**

5) **Conclusions**

6) **Future work**

7) **External resources**

1) Requirements

- The goal of the project is to create an application that allows:
  a) marking of an object on the image of the first camera
  b) sending that mark to the second camera
  c) second camera should be able to identify an object on its image

- Cameras will look on the same scene but from different positions

- The distance between cameras should be at least 10-20 meters

- As a main platform for the project an Android OS should be used

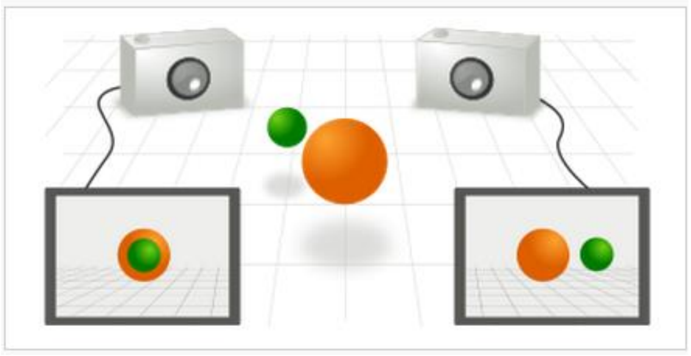
- OpenCV free library will be used for image processing algorithms

- Relative pose knowledge of the Android device will be used for improving the results of the image processing algorithms

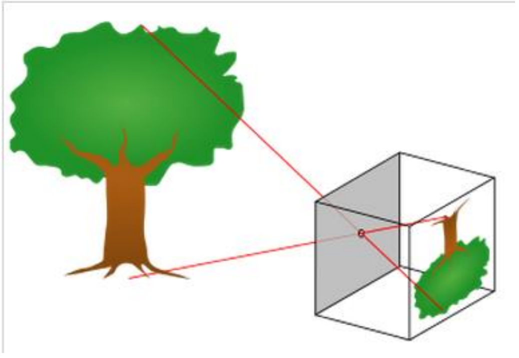
2) Overview

a) Epipolar geometry

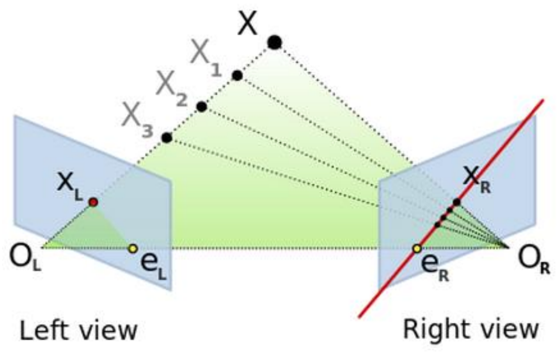
- Computer stereo vision is a part of the field of computer vision. Two cameras take pictures of the same scene, but they are separated by a distance – exactly like our eyes. A computer compares the images while shifting the two images together over top of each other to find the parts that match.
- Epipolar geometry is the geometry of stereo vision. When two cameras view a 3D scene from two distinct positions, there are a number of geometric relations between the 3D points and their projections onto the 2D images that lead to constraints between the image points. These relations are derived based on the assumption that the cameras can be approximated by the pinhole camera model.



- The pinhole camera model describes the mathematical relationship between the coordinates of a 3D point and its projection onto the image plane of an ideal pinhole camera, where the camera aperture is described as a point and no lenses are used to focus light. The model does not include, for example, geometric distortions or blurring of unfocused objects caused by lenses and finite sized apertures



- If the relative translation and rotation of the two cameras is known, the corresponding epipolar geometry leads to two important observations:



-

- If the projection point x_L is known, then the epipolar line $e_R – x_R$ is known and the pointX projects into the right image, on a pointx_R which must lie on this particular epipolar line. This means that for each point observed in one image the same point must be observed in the other image on a known epipolar line. This provides anepipolar constraint which corresponding image points must satisfy and it means that it is possible to test if two points really correspond to the same 3D point. Epipolar constraints can also be described by the essential matrix or the fundamental matrixbetween the two cameras.

- If the points x_L and x_R are known, their projection lines are also known. If the two image points correspond to the same 3D point X the projection lines must intersect precisely at X. This means that X can be calculated from the coordinates of the two image points, a process called triangulation.

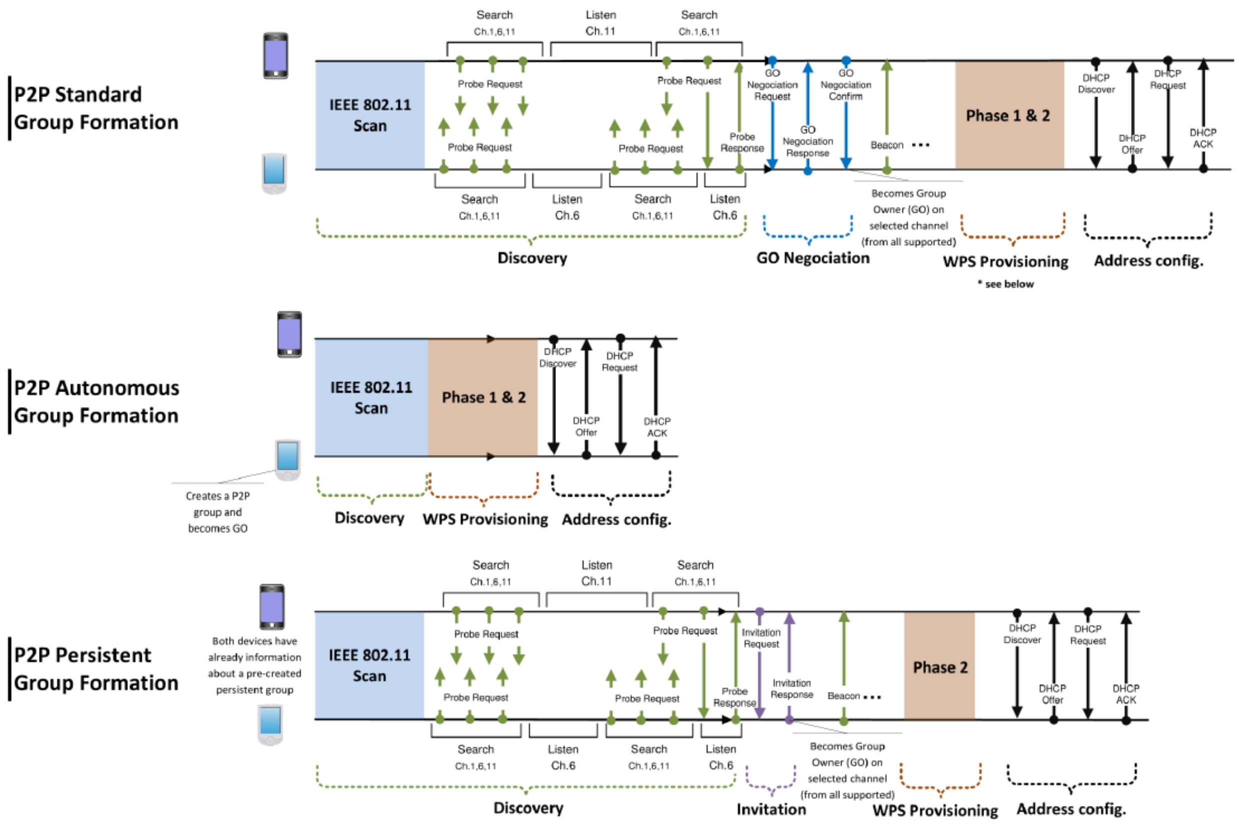
- Fundamental matrix $\mathbf{F}$ is a 3×3 matrix which relates corresponding points in stereo images. In epipolar geometry, with homogeneous image coordinates, xand x′, of corresponding points in a stereo image pair, F*x describes a line (an epipolar line) on which the corresponding point x′ on the other image must lie. That means, for all pairs of corresponding points holds

$$\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0.$$

b) Wi-Fi direct

- Wi-Fi Direct is a new technology defined by the Wi-Fi Alliance and aimed at enhancing direct device to device communications in Wi-Fi (without requiring the presence of the **explicit** access point). This technology can be entirely implemented in software over traditional Wi-Fi technology.
- On android devices Wi-Fi direct is available since the Android 2.4.
- Wi-Fi direct is built upon IEEE 802.11 infrastructure mode and lets devices negotiate who will take over the functionalities of the AP (one of the devices becomes an **implicit** AP, sometimes called **soft-AP**).
- Wi-Fi direct devices establish P2P Group which functions similarly to the regular Wi-Fi infrastructure mode. The device which acts as a soft-AP called **Group Owner**, other devices are called **P2P clients.**

- During initial connection 2 devices negotiate about the roles: one of them will be **Group Owner** and other one **P2P client**. After that any additional device can join the group as a **P2P client**.
- There are 3 possible Group Formations:
    1. **Standard**
       Discovery (sending and listening to probes) -> GO Negotiation (deciding who will be GO) -> WPS Provisioning (security configuration exchange between devices) ->DHCP exchange(IP address configuration)
    2. **Autonomous** (one of the devices sets itself as a GO)
       Discovery -> WPS Provisioning ->DHCP exchange(IP address configuration)
    3. **Persistent** (in this way, the devices forming the group store network credentials andthe assigned P2P GO and Client roles for subsequent re-instantiations of the P2P group)
       Discovery -> Invitation -> WPS Provisioning ->DHCP exchange(IP address configuration)



- The P2P GO is also required to run aDynamic Host Configuration Protocol (DHCP) server toprovide P2P Clients with IP addresses (in our case it is done automatically by Android)
- Also Wi-Fi Direct **does not allow transferring the role of P2P GO** within a P2P Group. In this way, if the P2P GOleaves the P2P Group then the

group is torn down, and hasto be re-established using some of the specified procedures.

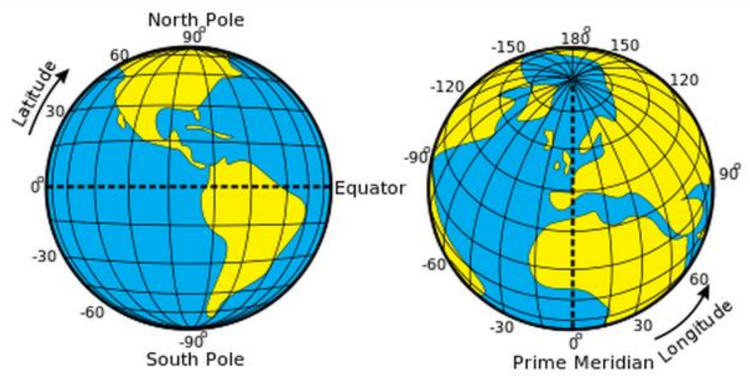
- This causes some difficulties during implementation of the Wi-Fi Direct support on Android OS. **The only IP which is available through Android OS API (after initial negotiation) is an IP of the GO**. As a result GO won't be able to send messages to the P2P client (because its IP cannot be known to the GO through the Android OS API). To overcome this difficulty in our app a P2P client should get its IP and send it to the GO, after that GO will be able to send messages to the P2P.

c) Android sensors

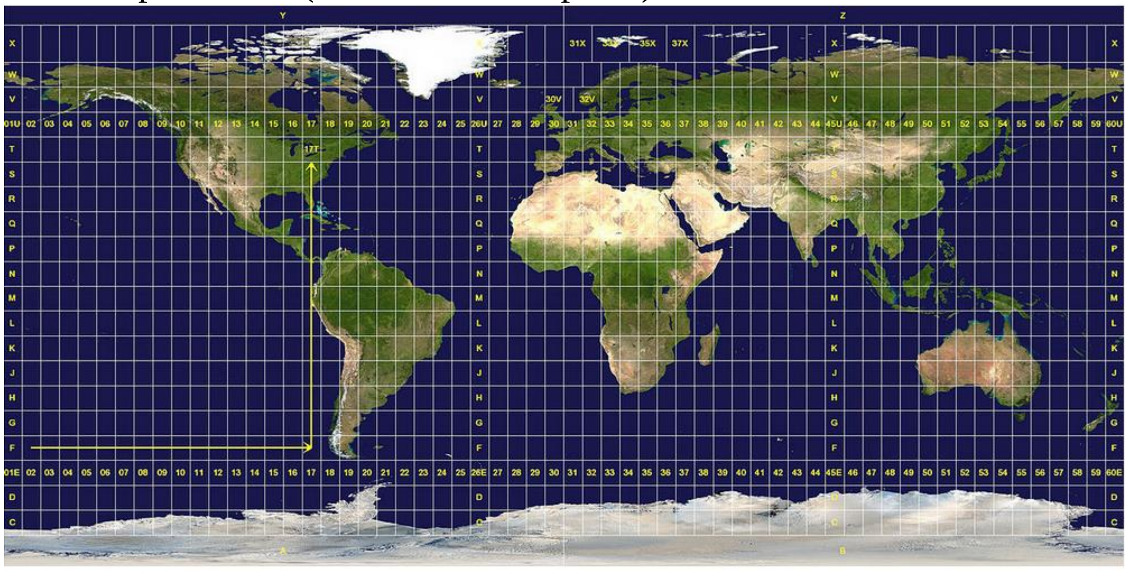
To improve the result of the points matchingfor some algorithms we use location and orientation data

1. Location – android API supplies us with information about latitude, longitude and altitude (using WGS84 geodetic datum). Using this information we convert geographic coordinates to the UTM (Universal Transverse Mercator). After that UTM coordinates will be supplied to SOREPP fundamental matrix estimator.
2.

Latitude, longitude explanations:

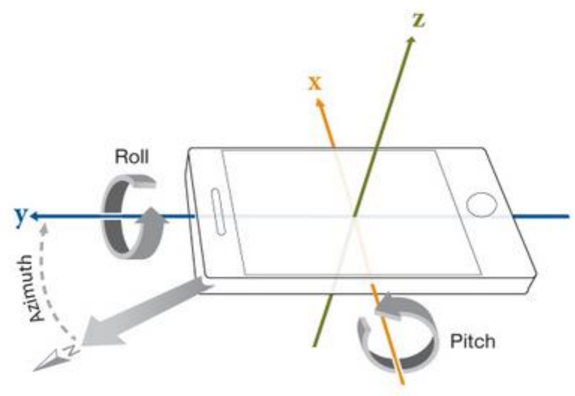


UTM map is below (taken from Wikipedia):

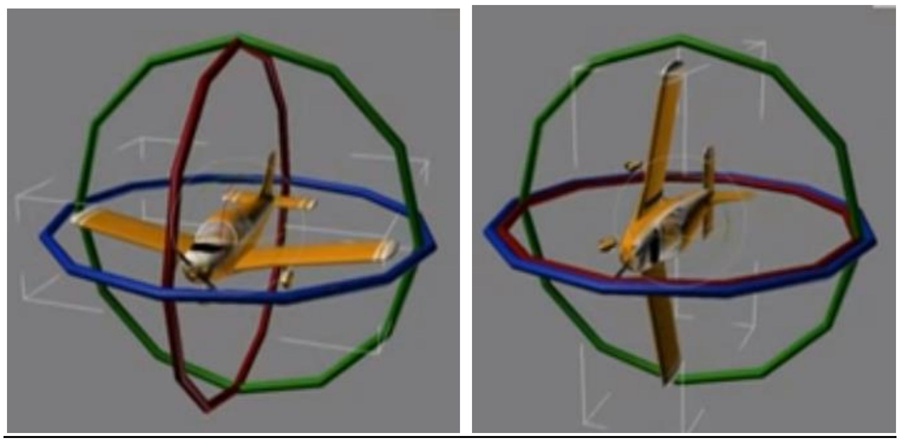


3. Orientation – android API supplies us with information about azimuth pitch and roll. That information is adjusted to the coordinate ranges found in GeoCam application. After that an information can be passed to SOREPP for improving the results of finding the fundamental matrix.

   Note:you should hold your device in front of your face to get the correct orientation, if you perform roll of 90 degrees (for example when your android lies on the table) you may cause Gimbal lock as a result of that the orientation data will be incorrect



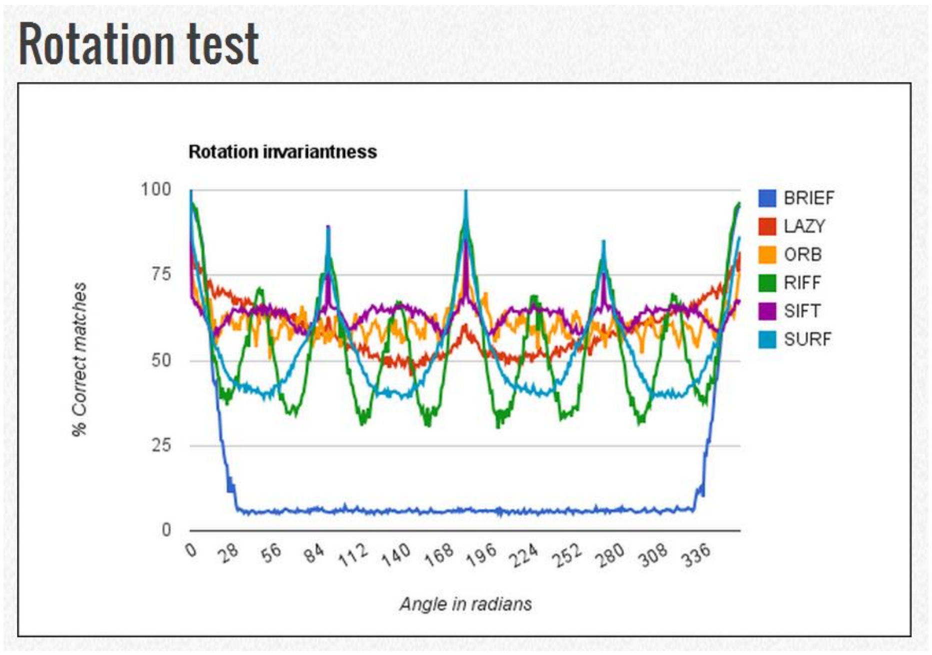
- Gimbal lock is the loss of one degree of freedom in a three-dimensional mechanism that occurs when the axes of two of the three gimbals are driven into a parallel configuration, "locking" the system into rotation in a degenerate two-dimensional space. On the picture below you can see the gimbal lock which is caused by roll of 90 degrees.

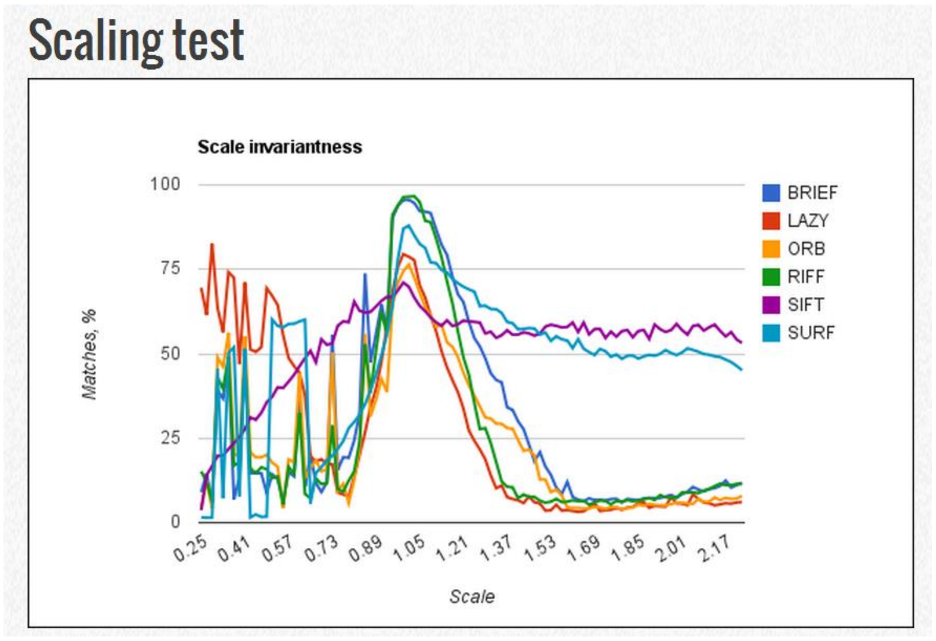


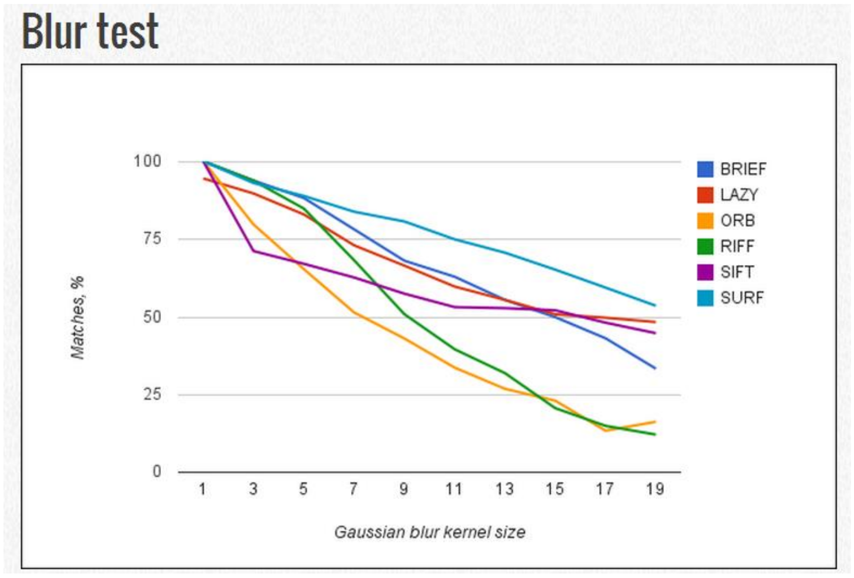
d) Algorithms for points matching

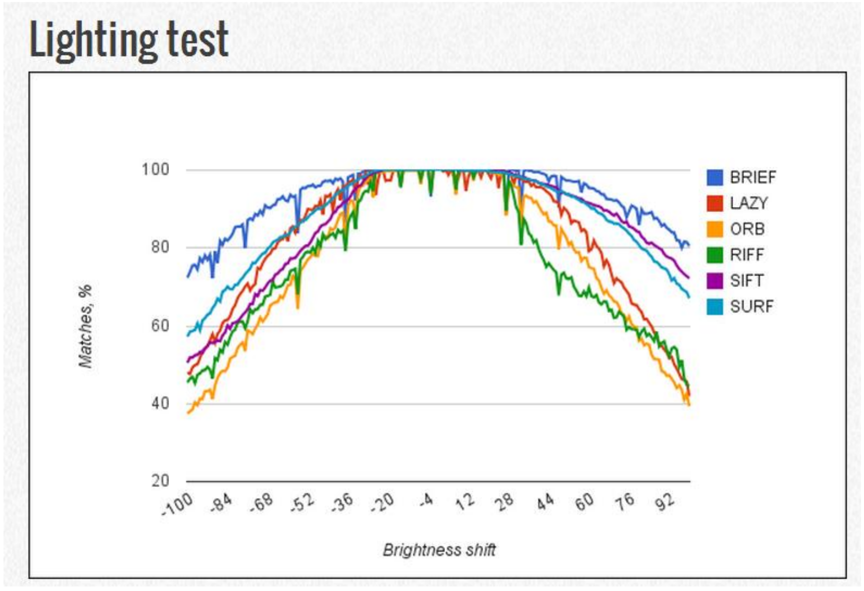
   Contemporary features matching algorithms such as SIFT and SURF able to locate corresponding points that contain points of interest such as edges and angles. However, they are not able to find a matching location of a random chosen point. Our project aims to solve this problem by search for corresponding points based on matched features and relative pose knowledge.

As a main feature detector and extractor we used SIFT. The following diagrams show its superiority over other detectors and extractors (diagrams were taken from the site which link appears in external links section)







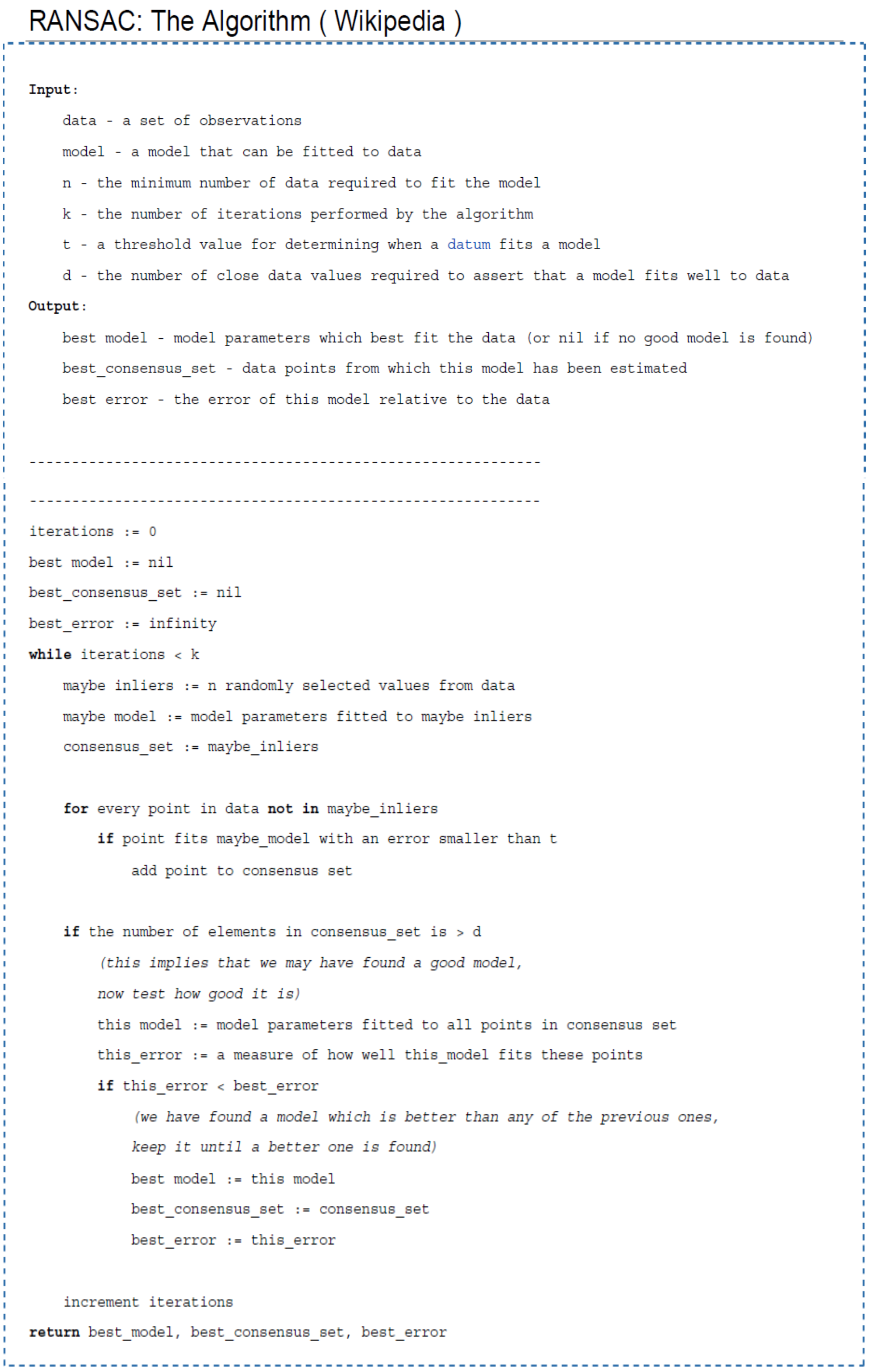


The following approaches were used to solve the problem:

1. <u>Based on fundamental matrix:</u>

- For fundamental matrix estimation we used:
  1) SOREPP (Robust Epipolar Geometry Estimation using Noisy Pose Priors). This algorithm uses data from orientation and location sensors of the device and according to the authors should outperform RANSAC. <u>**Algorithm was developed at the Technion by Yehonatan Goldman**</u>
  2) RANSAC (RANdom SAmple Consensus). Iterative method to estimate parameters of a mathematical model (x1*F*x=0) from a set of observed data (found by SIFT) which may contain outliers. The following pseudo-

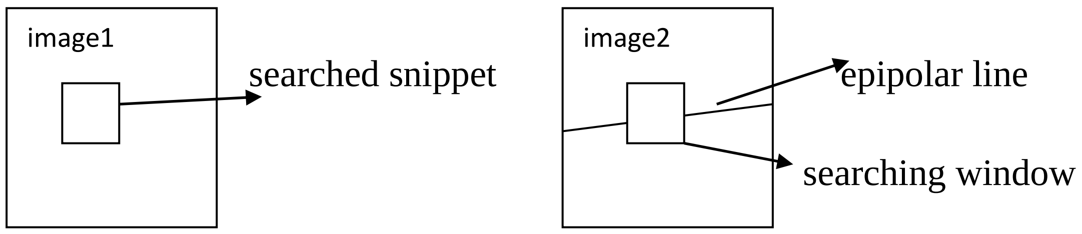
algorithm is used for fundamental matrix estimation (taken from wikipedia):



RANSAC: The Algorithm (Wikipedia)

```
Input:
    data - a set of observations
    model - a model that can be fitted to data
    n - the minimum number of data required to fit the model
    k - the number of iterations performed by the algorithm
    t - a threshold value for determining when a datum fits a model
    d - the number of close data values required to assert that a model fits well to data
Output:
    best model - model parameters which best fit the data (or nil if no good model is found)
    best_consensus_set - data points from which this model has been estimated
    best error - the error of this model relative to the data

------------------------------------------------------------
------------------------------------------------------------
iterations := 0
best model := nil
best_consensus_set := nil
best_error := infinity
while iterations < k
    maybe inliers := n randomly selected values from data
    maybe model := model parameters fitted to maybe inliers
    consensus_set := maybe_inliers

    for every point in data not in maybe_inliers
        if point fits maybe_model with an error smaller than t
            add point to consensus set

    if the number of elements in consensus_set is > d
        (this implies that we may have found a good model,
        now test how good it is)
        this model := model parameters fitted to all points in consensus set
        this_error := a measure of how well this_model fits these points
        if this_error < best_error
            (we have found a model which is better than any of the previous ones,
            keep it until a better one is found)
            best model := this model
            best_consensus_set := consensus_set
            best_error := this_error

    increment iterations
return best_model, best_consensus_set, best_error
```

- For location matching across epipolar line we used the following approaches
  1) SSD - sum of squared differences is used between searched snippet and the snippets across an epipolar line. We use the following formula to estimate how different the snippets are: SSD $= \sum\sum(I_{left} - I_{right})^2$



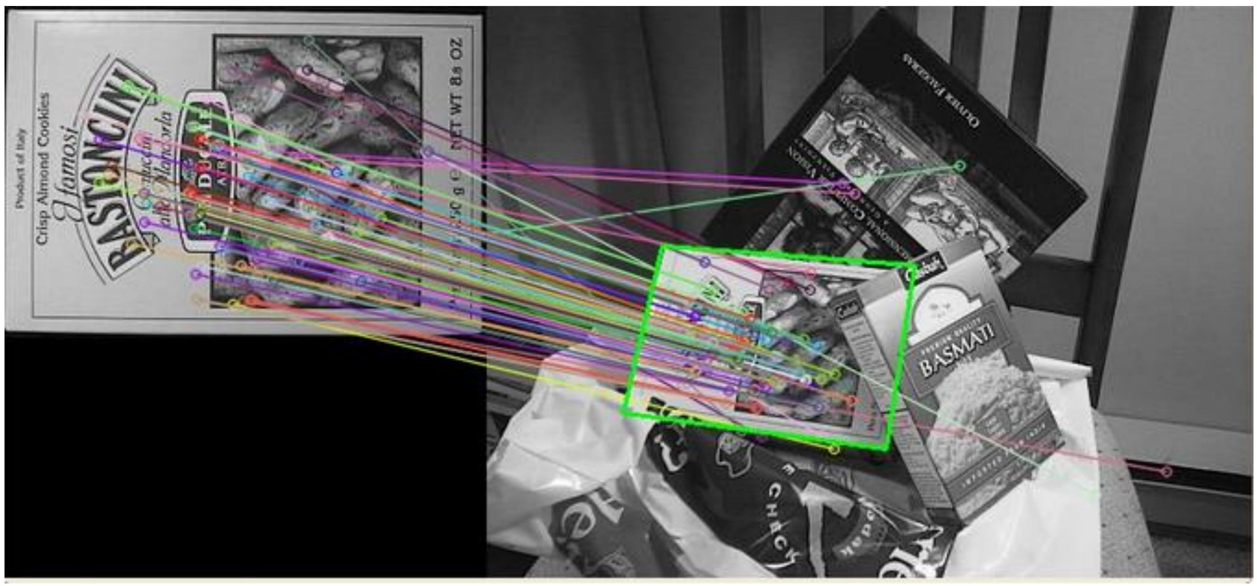
  2) aHash (also called average hash, it is used between searched snippet and the snippets across an epipolar line). Process is very similar to the process of calculating SSD across epipolar line but instead of SSD we calculate 64 bit aHash for the snippet. The following algorithm is applied:
     - resize the snippet into a grayscale 8x8 image
     - calculate the average value of the pixels from 8x8 image
     - calculate a hash from the 8x8 image (if the intensity of the pixel is bigger than the average then set bit to 1 otherwise to 0)

  3) pHash(also called "Perceptive Hash"). This algorithm is similar to aHash but uses a discrete cosine transform (DCT) and compares based on frequencies rather than color values.The following algorithm is applied:
     - resizes an image to 32x32
     - apply DCT and get 32x32 matrix of DCT coefficients
     - get only top left 8x8 block from matrix (which contains low frequencies, high frequencies are responsible for details of the picture)
     - calculate the average intensity of 8x8 block
     - calculate a hash from the 8x8 top left block (if the intensity of the pixel is bigger than the average then set bit to 1 otherwise to 0)

  4) ThreePoints matcher (tries to find target based on the 3 feature points closest to the chosen point). The following algorithm is applied:
     - find the 3 feature points closest to the searched point which were previously found by SIFT (during feature detection and extraction)
     - calculate transformation matrix using the 3 points found before and their matches on second picture
     - find the target point x using the transformation matrix above

2. Based on homography:

- In the field of computer vision, any two images of the same planar surface in space are related by a homography. (Image example is taken from openCV homography tutorial)



- RANSAC is used for finding homography matrix.
- Contrast is the difference in luminance and/or color that makes an object
- If contrast is very small (most of the pixels in the histogram can be located in a very small region) then we can assume that the region of interest is a plane and use an algorithm based on the homography otherwise we will use an approach based on the fundamental matrix (SSD for example)

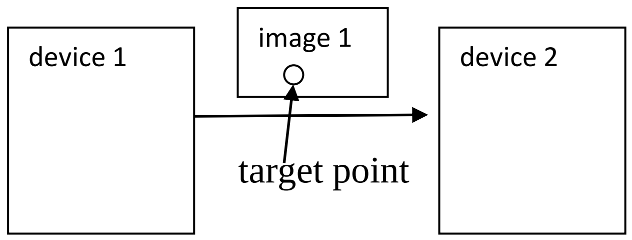
3. Competitive approach

- All algorithms described above can run together to find a number of possible solutions (targets) on the image 2.
- From fundamental matrix and solutions found above we calculate epipolar lines back on the image 1
- An epipolar line which is the closest to the original point on image 1 determine the winner

e) Different resolutions

- Different resolutions usually result in different density of pixels which may cause the problem during SSD, aHash and pHash because of searching window which may hold more information on the screen with low pixels density.
- To avoid the possible problems some transformations are needed before running algorithms.

- Let's assume that image 1 with target information is sent from device 1 to device 2 which contains image 2.



- In this case two cases are possible:
    - **image 1 is bigger than an image 2** (in this case we should decrease the size of image 1 to the size of the image 2 and adjust the target point and searching window size on it accordingly). Scale coefficients are calculated as follows:

      heightScaleCoefficient = image2.height/image1.height (<1)
      widthScaleCoefficient = image2.width/image1.width (<1)
      image1TargetAdjusted.X = image1Target.X * heightScaleCoefficient
      image1TargetAdjusted.Y = image1Target.Y * widthScaleCoefficient
      searchingWindowSizeAdjusted = searchingWindowSize * widthScaleCoeff

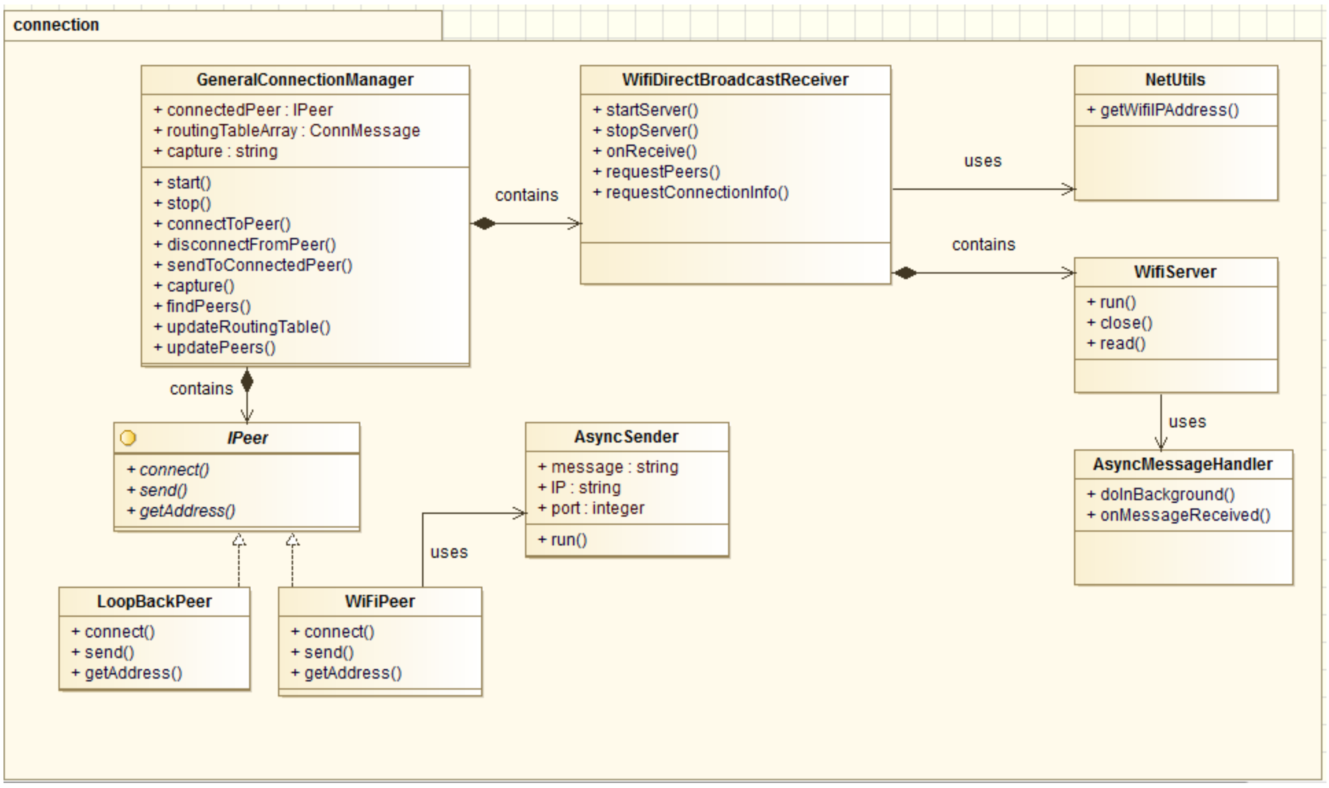
    - **image 2 is bigger than image 1** (in this case we should decrease the size of the image 2 and after finding the target on it we should scale target coordinates and epipolar line coefficients a,b,c accordingly (epipolar line is expressed in opencv as ax+by+c=0))

      heightScaleCoefficient = image2.height/image1.height (>1)
      widthScaleCoefficient = image2.width/image1.width (>1)
      image2FoundTargetAdjusted.X = image2Target.X * heightScaleCoefficient
      image2FoundTargetAdjusted.Y = image2Target.Y * widthScaleCoefficient
      epipolarLineAdjustedA = epipolarLineA * heightScaleCoefficient
      epipolarLineAdjustedB = epipolarLineB * widthScaleCoefficient
      epipolarLineAdjustedC = epipolarLineC * heightScaleCoefficient * widthScaleCoefficient

(calculations are done based on the epipolar line formula a*x + b*y + c = 0 and transformations xAdjusted = x*widthScaleCoefficient, yAdjusted = y*heightScaleCoefficient)

3) Design

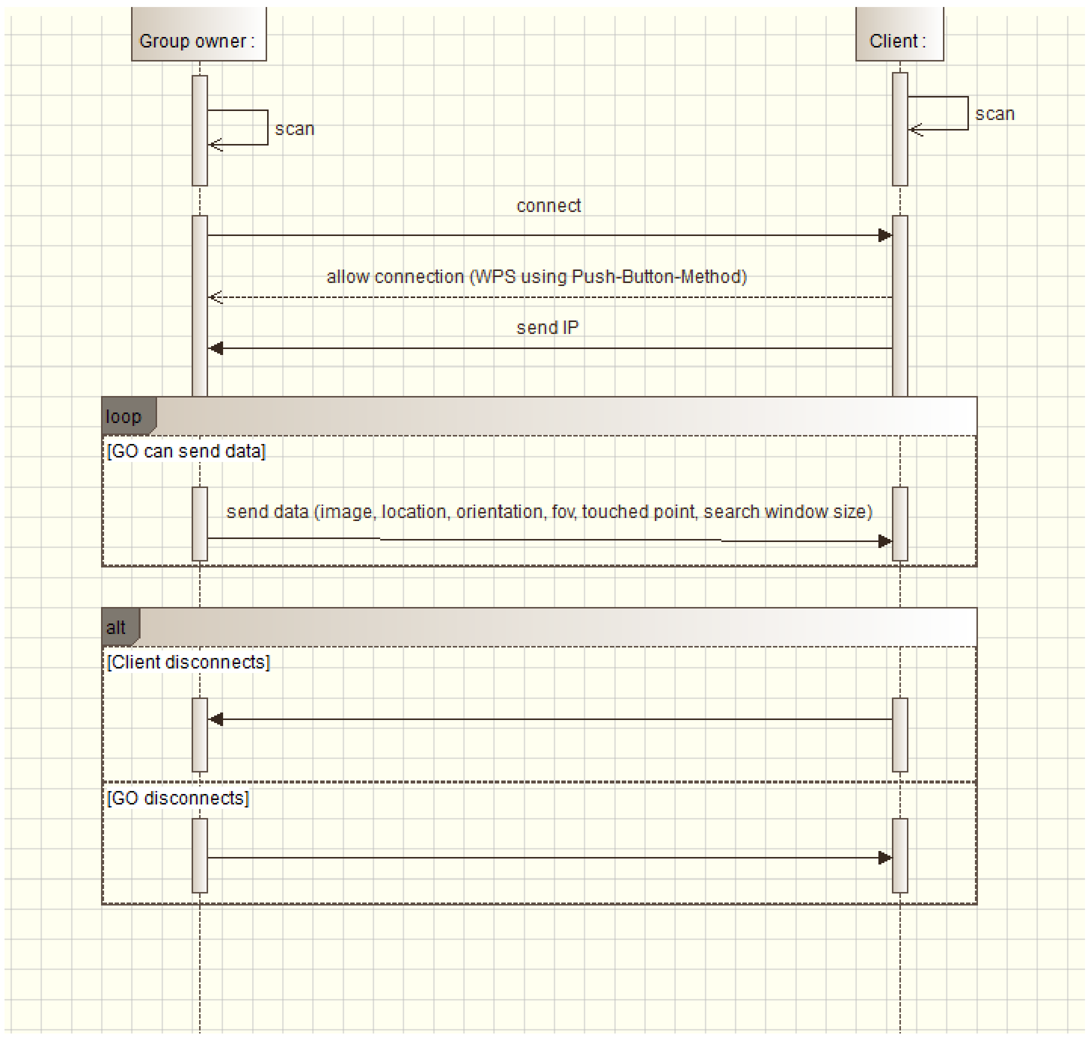
a) **<u>connection package</u>** – contains the connection service which allows to send and receive messages via WiFi direct android interface



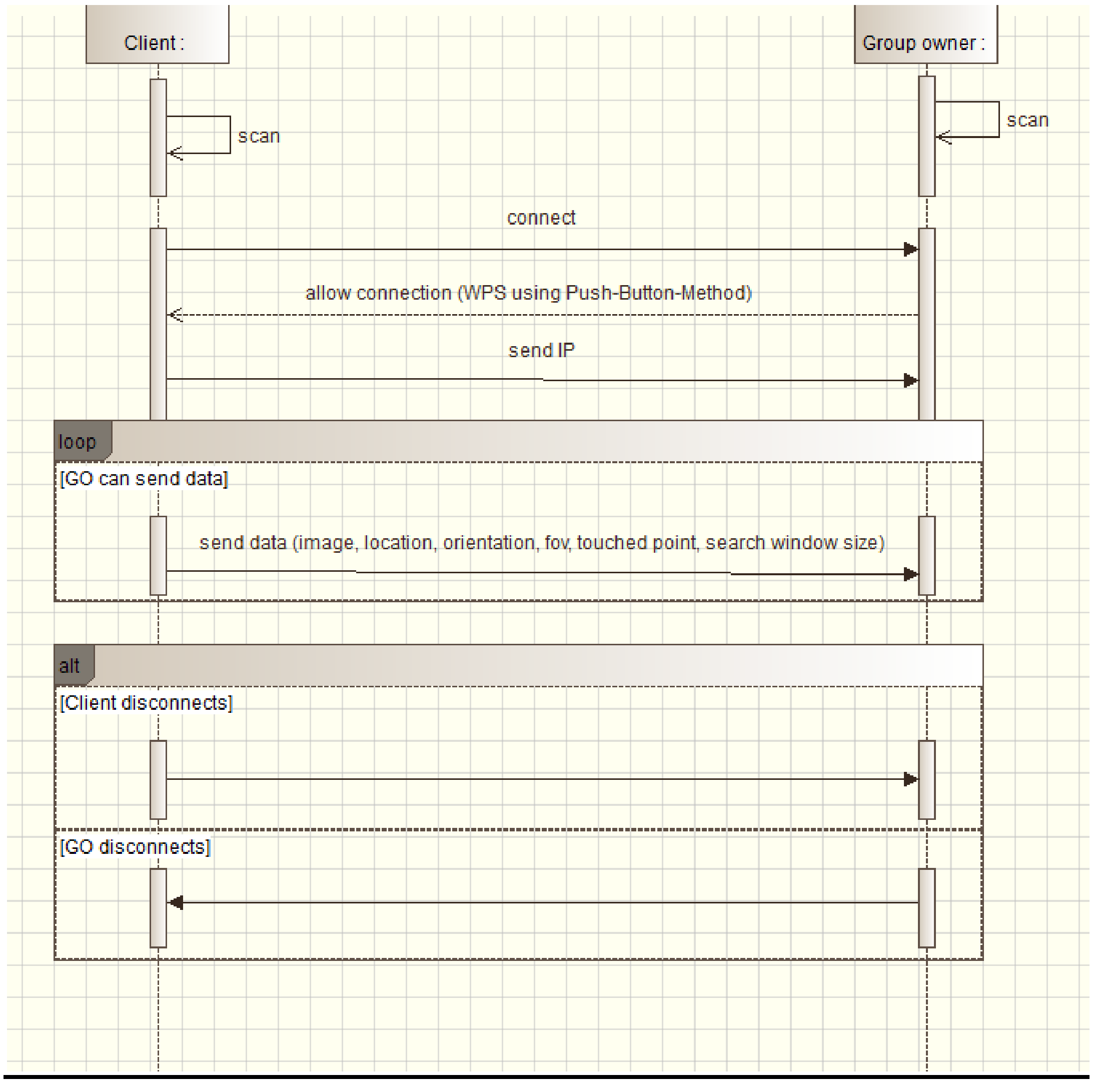
<u>Note:</u> As can be seen in the diagram there are 2 peers WifiPeer and LoopBackPeer. <u>WifiPeer</u> is used for connection between 2 physical devices and <u>LoopBackPeer</u> is used when you have only 1 physical device. LoopBackPeer allows device to capture and send message to itself. This is very useful for testing of the new algorithms

Mainly there are two use cases available for the users for communication between devices:

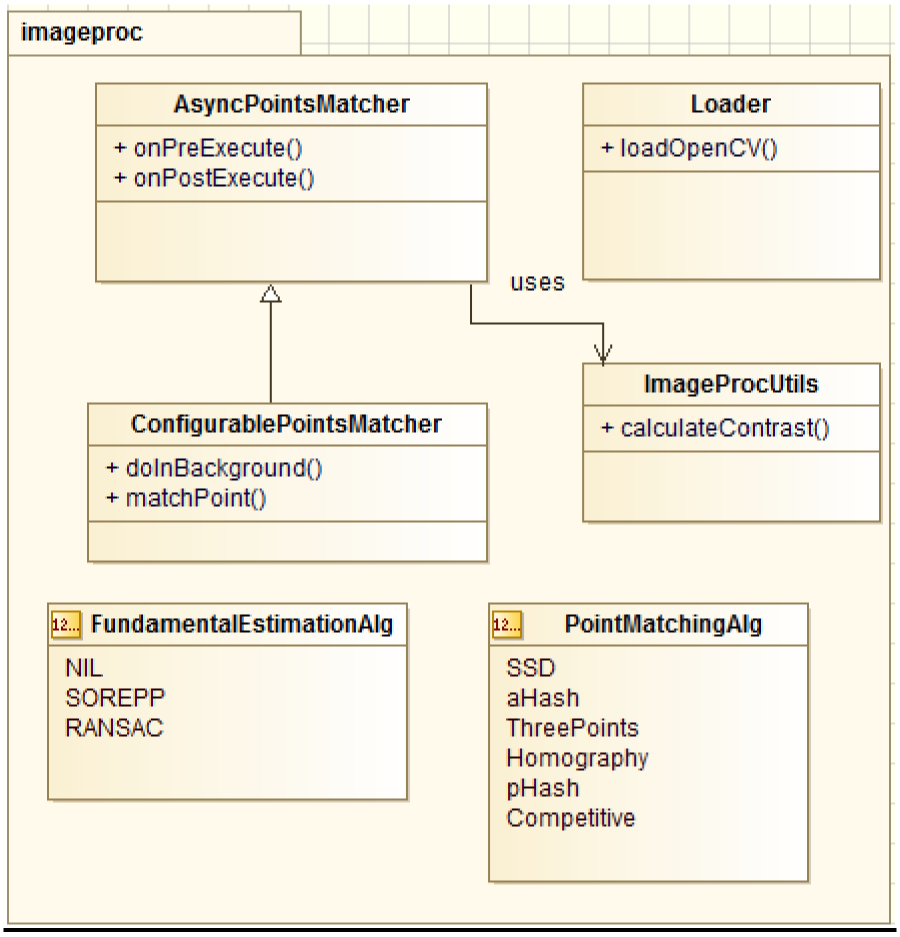
1. <u>From GO (group owner) to the client</u>. As you can see in the sequence diagram below GO tries to connect to the client and after that sends him/her the information about the target (image, touched point, fov, orientation, location). At the end any part of the communication can initiate disconnect. **<u>Pay attention</u>** that if any part attempts to scan wifi or connect to another peer then the current connection between GO and its client will be lost as well. Connection is available only in one way, it means that if GO connects to the client then **only** GO can send information to the client. If client wants to send target data to GO then client should disconnect from current connection and connect to the GO


Group owner :
Client :
scan
scan
connect
allow connection (WPS using Push-Button-Method)
send IP
loop
[GO can send data]
send data (image, location, orientation, fov, touched point, search window size)
alt
[Client disconnects]
[GO disconnects]

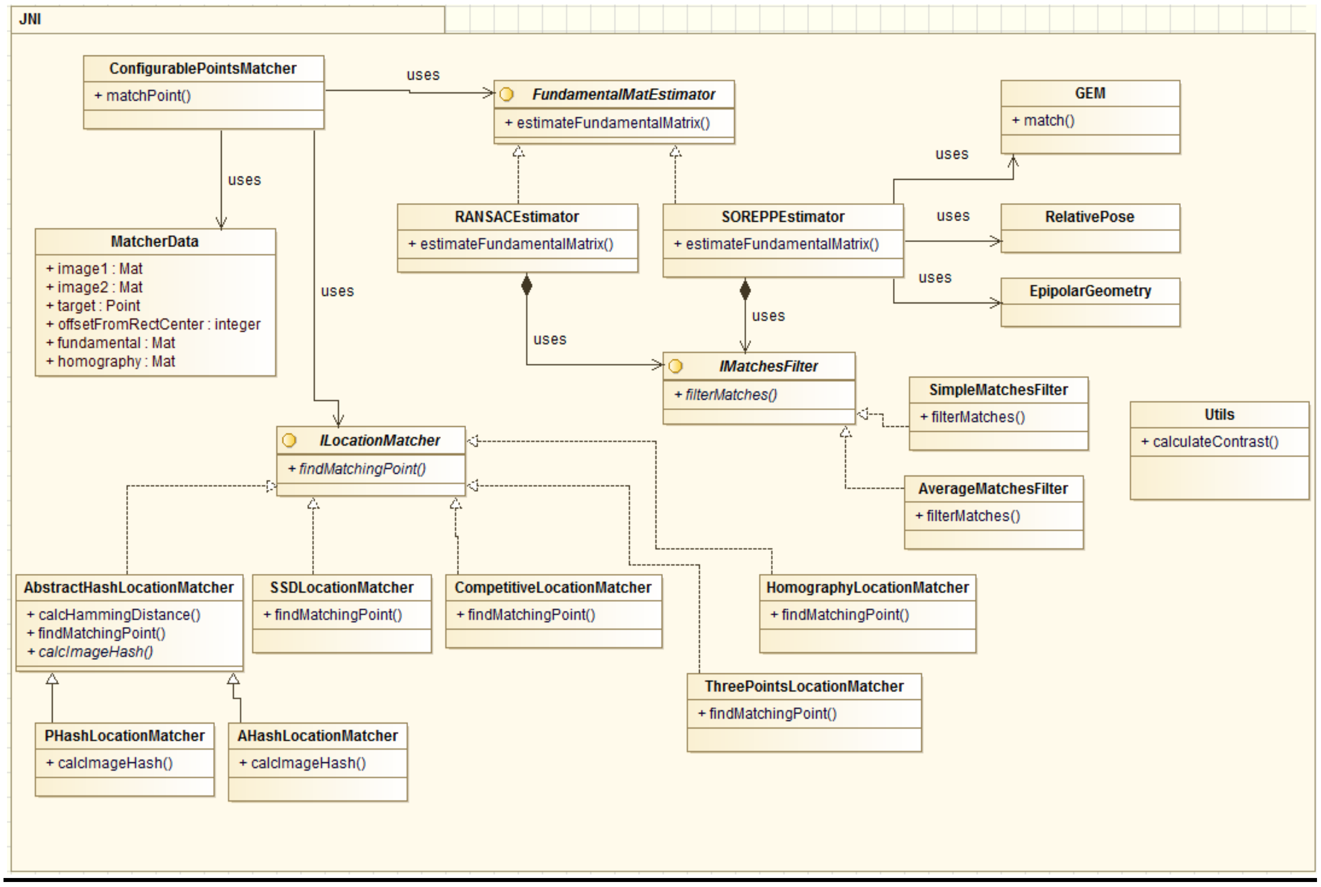
2. From client to GO. All rules are the same as before except that at this time client initiates connection and sends an information about the target to the GO


Client :
Group owner :
scan
scan
connect
allow connection (WPS using Push-Button-Method)
send IP
loop
[GO can send data]
send data (image, location, orientation, fov, touched point, search window size)
alt
[Client disconnects]
[GO disconnects]

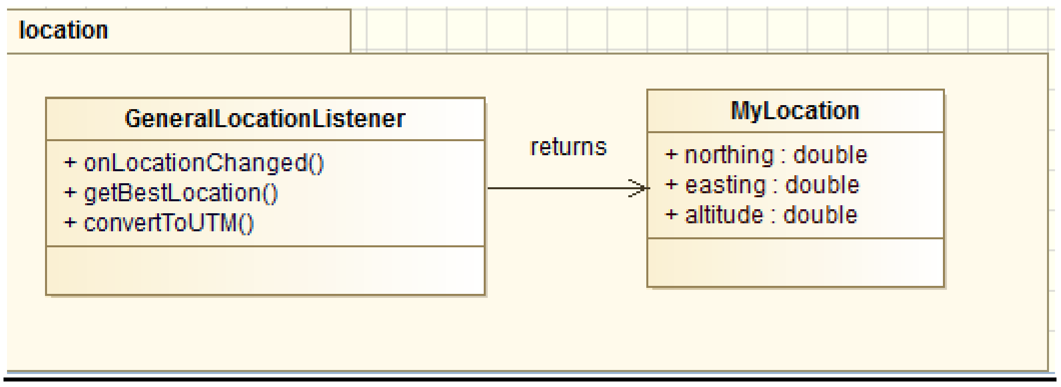
b) <u>imageproc package</u> – contains wrapper for JNI part of the project



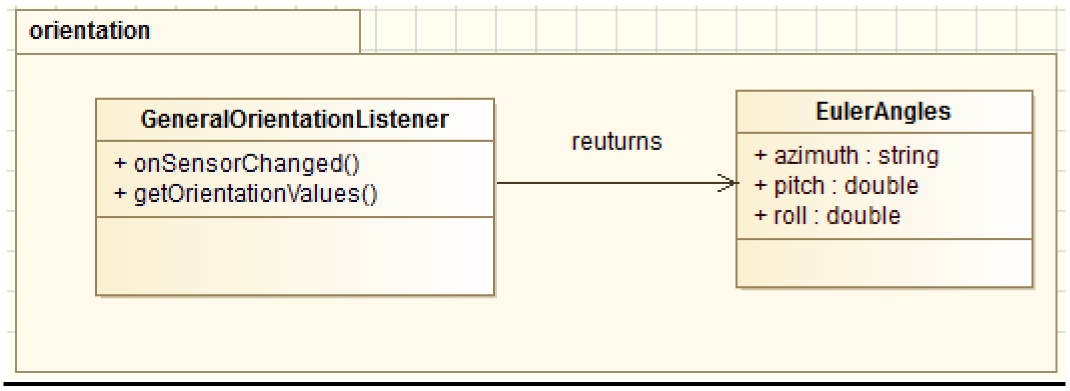
c) **JNI folder** – contains image processing algorithms (using OpenCV libraries)



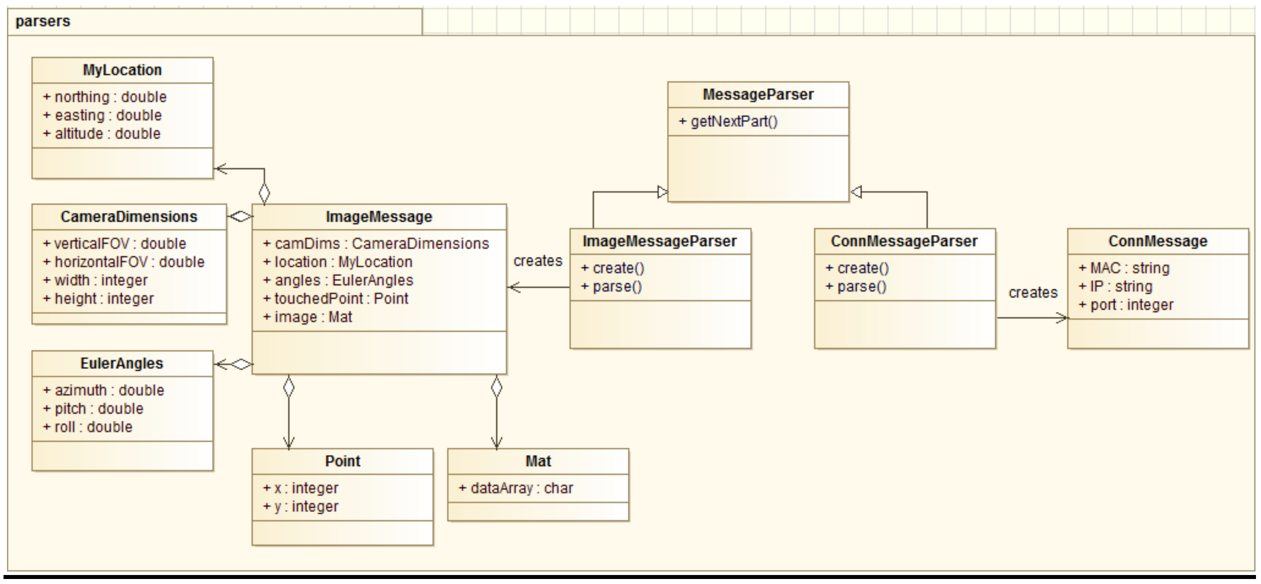
d) **location package** – contains android location listener which notifies about changes in the location of the device



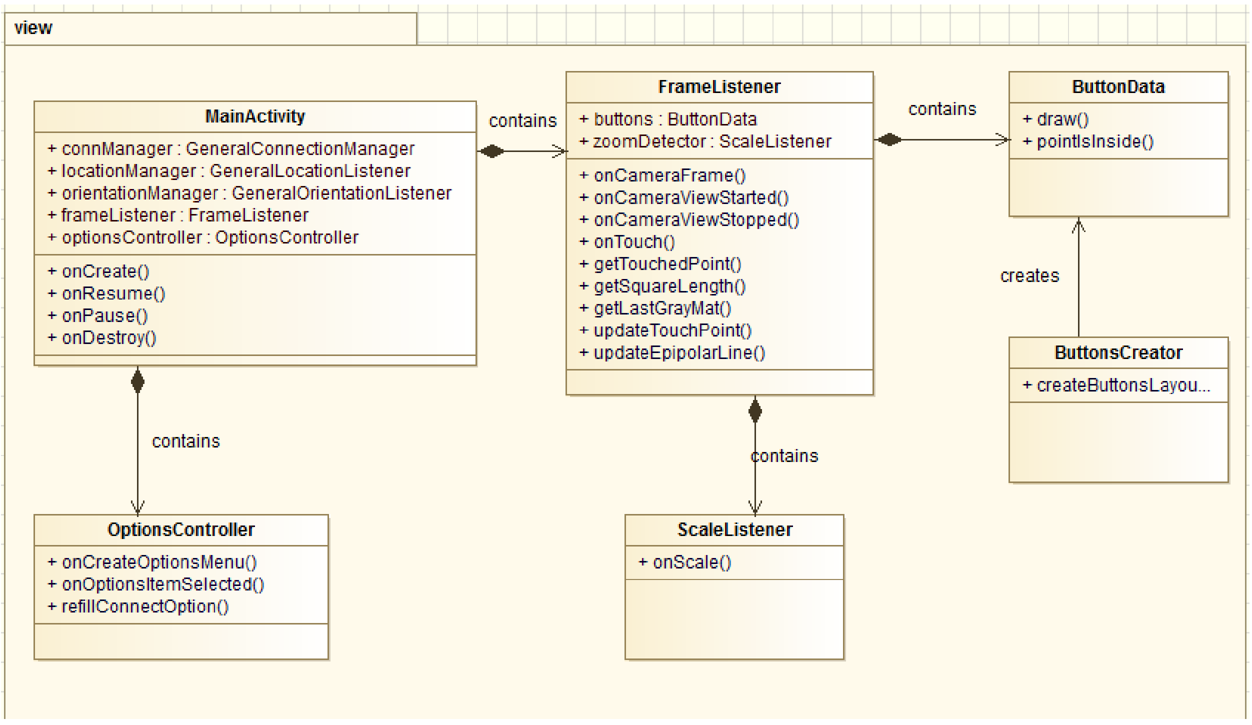
e) **<u>orientation package</u>** – contains orientation listener which notifies about changes in android orientations (we use azimuth, pitch and roll similar to the GeoCam android app).



f) **<u>parser package</u>** – contains classes which handle parsing and creating of the messages which are sent between android devices



g) **view package** – contains classes which allows user-device interaction (main activity, touch listener, etc.)



4) Testingresults

First test:

- To test our approaches we used 2sets of street pictures. One set contained 5 pictures and another one 3 pictures
- 10 random (x,y) coordinates were chosen
- Each picture in the set was tested against the other pictures in the set

Success rate which we received

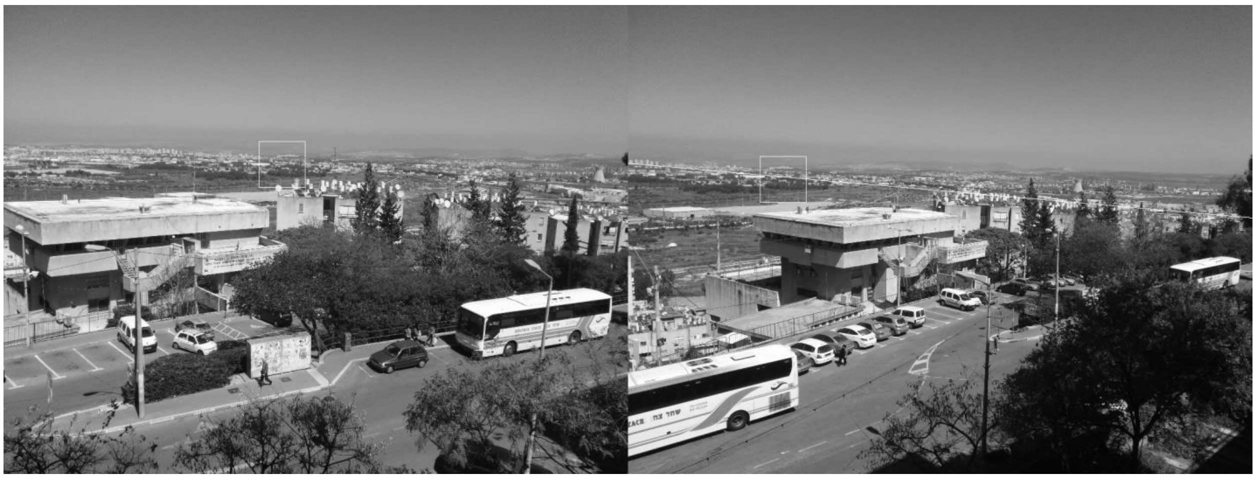
	RANSAC	**SOREPP**
SSD	**57/120 = 47.5%**	**66/120 = 55%**
aHash	**56/120 = 46.7%**	**57/120 = 47.5%**
Homography	**48/120 = 40%**	

Note: the points which were chosen in the sky or the points which cannot be found on another picture were not counted. For example the following resultswere not counted

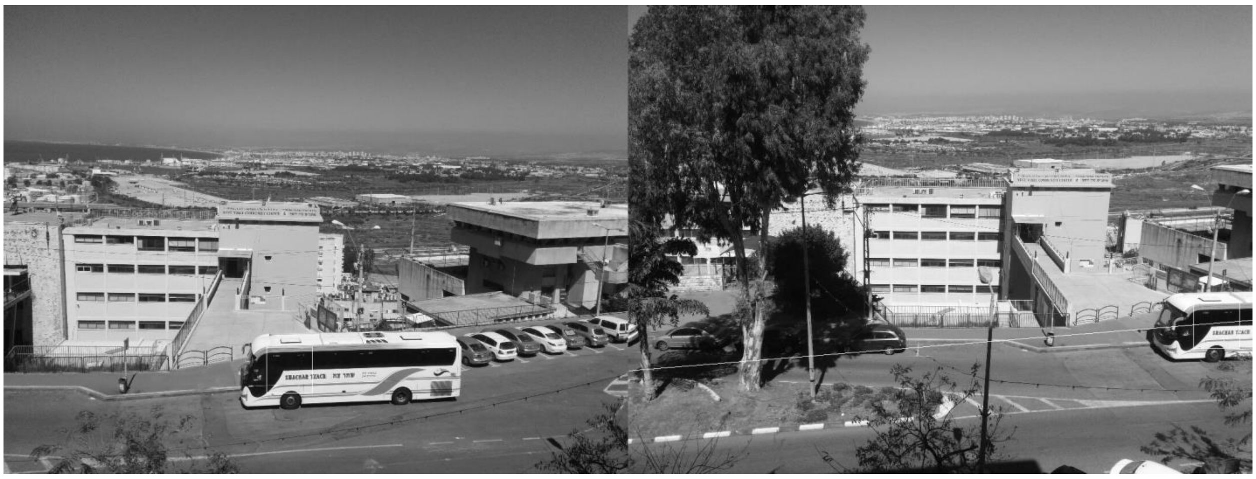
- Some interesting results:

  - RANSAC+aHash

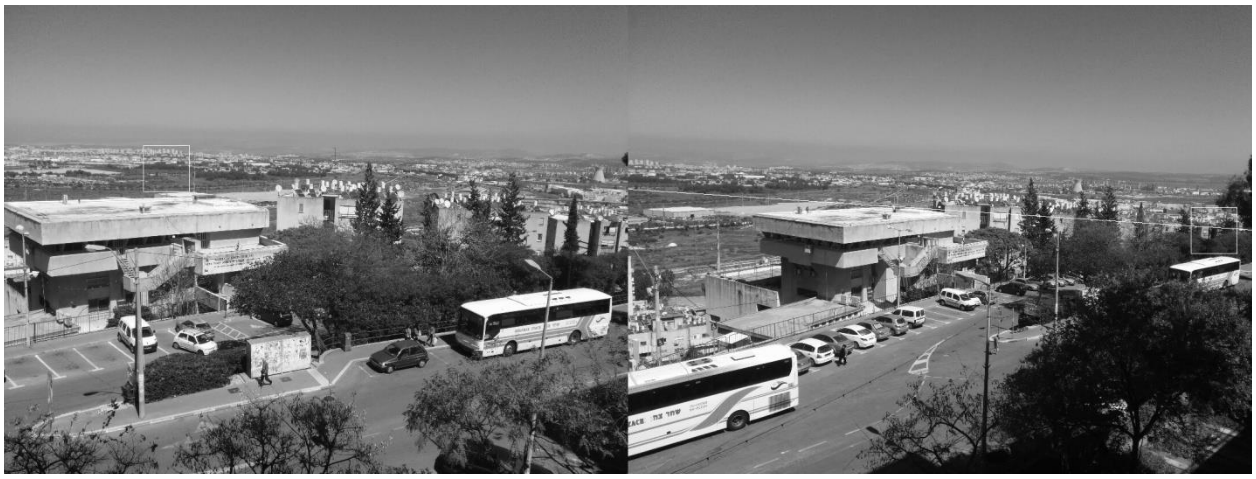
    Found far-away target



Found close target

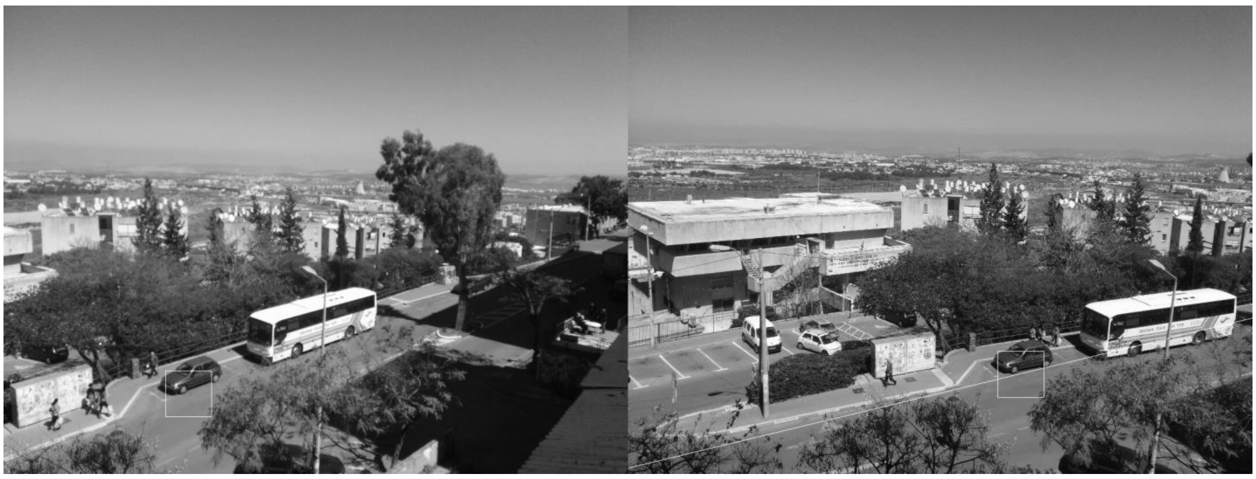


Didn't find far-away target even though an epipolar line is correct

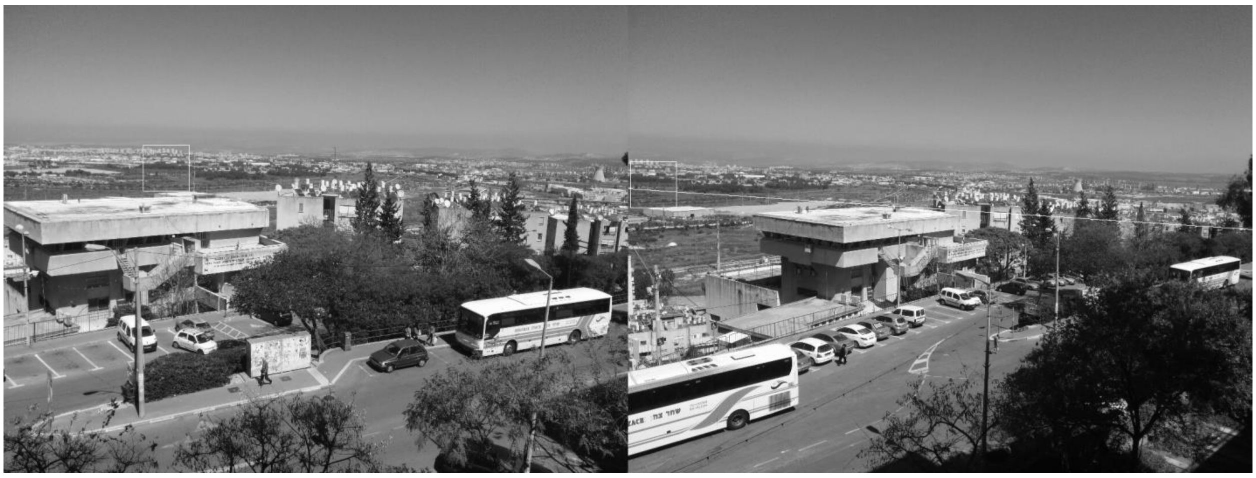


- RANSAC +SSD

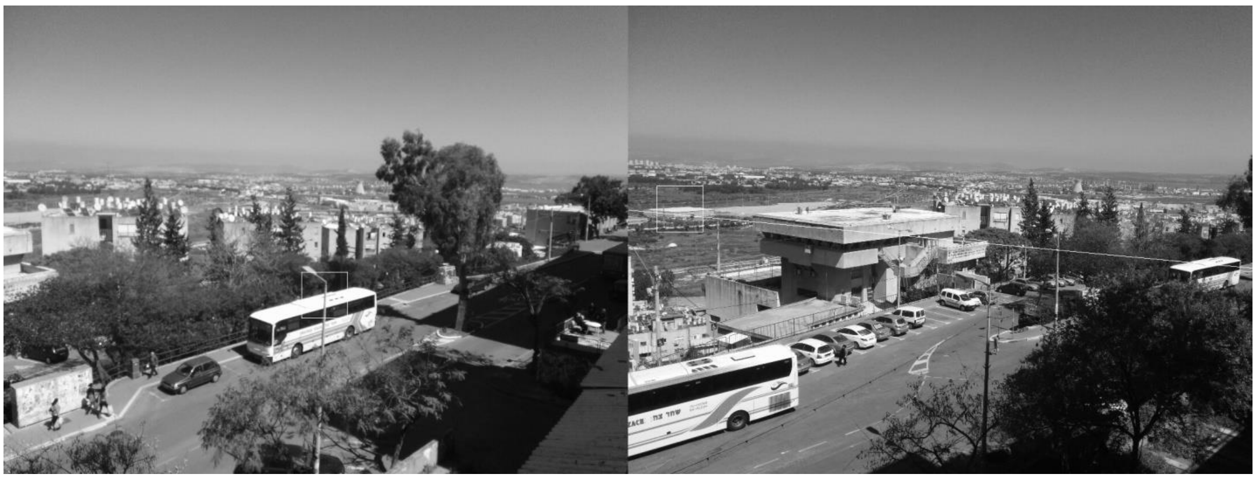
Found the car



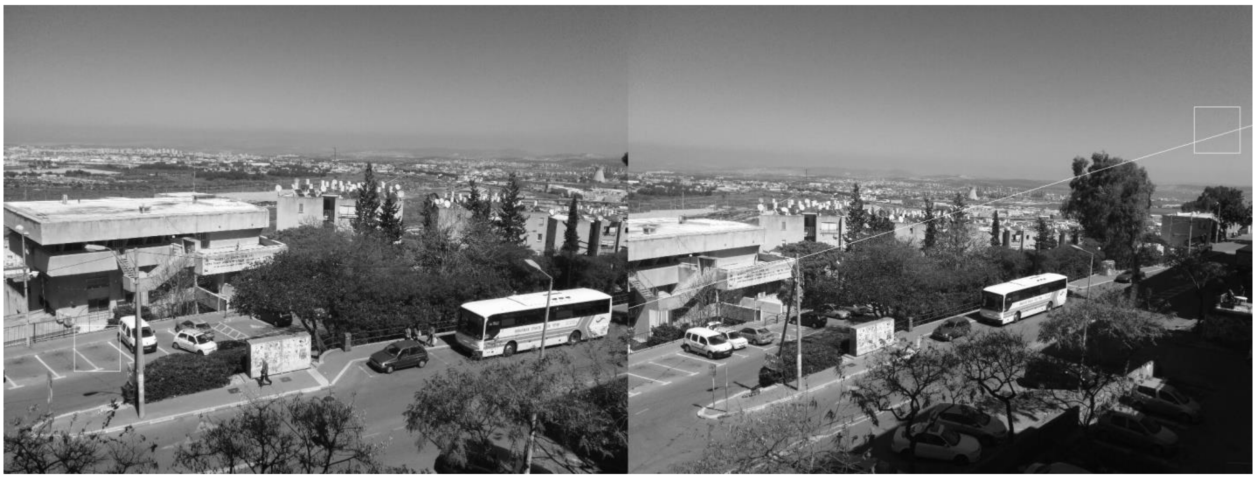
Found far-away target



Didn't find the target even though an epipolar line is correct

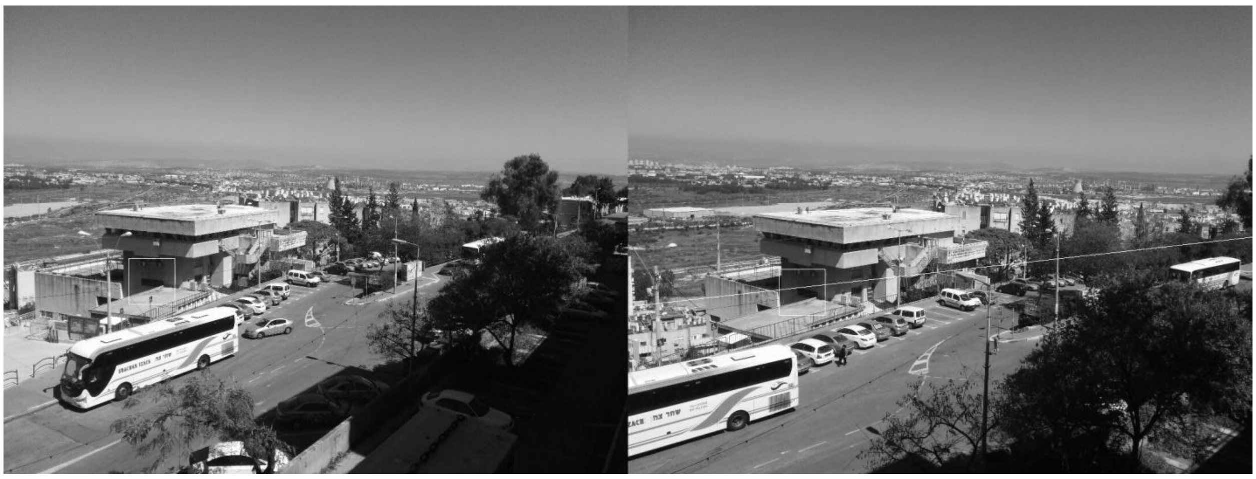


Didn't find (the case where a homography probably would work)

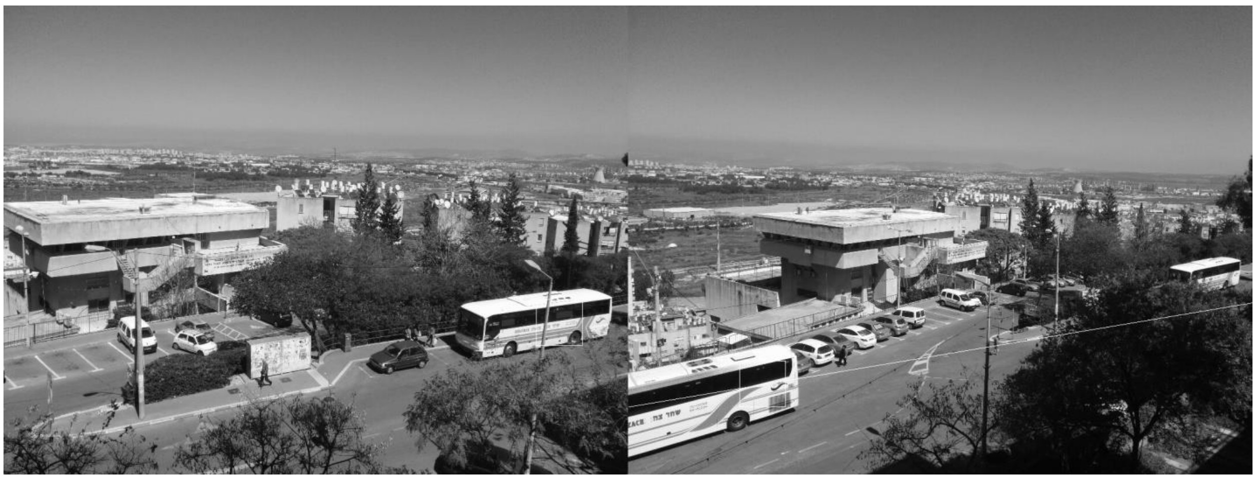


- SOREPP+aHash

Correct window was found additionaly we can observe very precise epipolar line

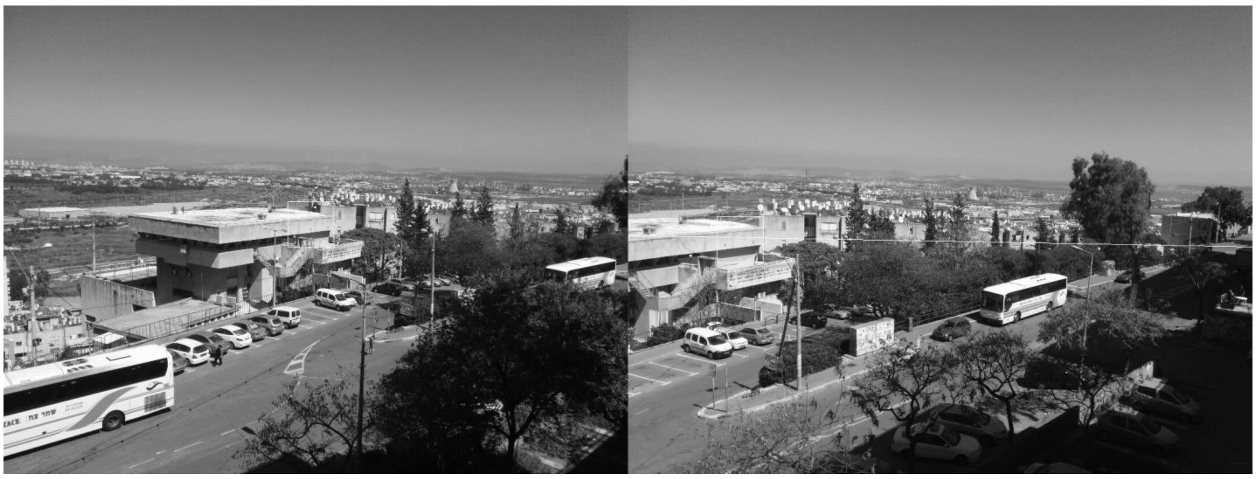


aHash found very similar target, but unfortunately incorrect bus, probably homography or 3Points approach would help

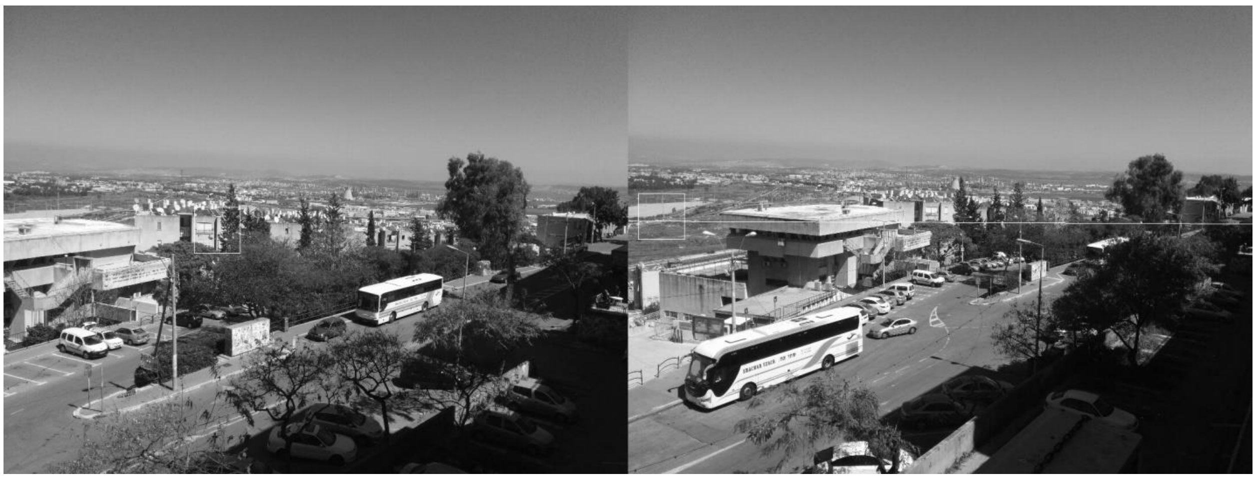


- SOREPP +SSD

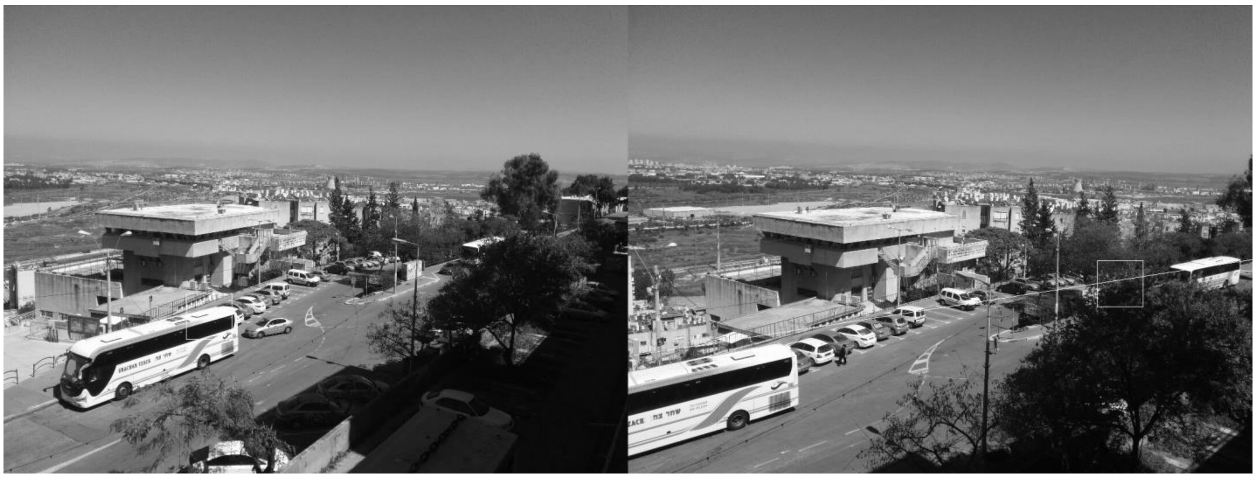
Roof angle of the building was found (note one more time the precise epipolar line)



SSD didn't work



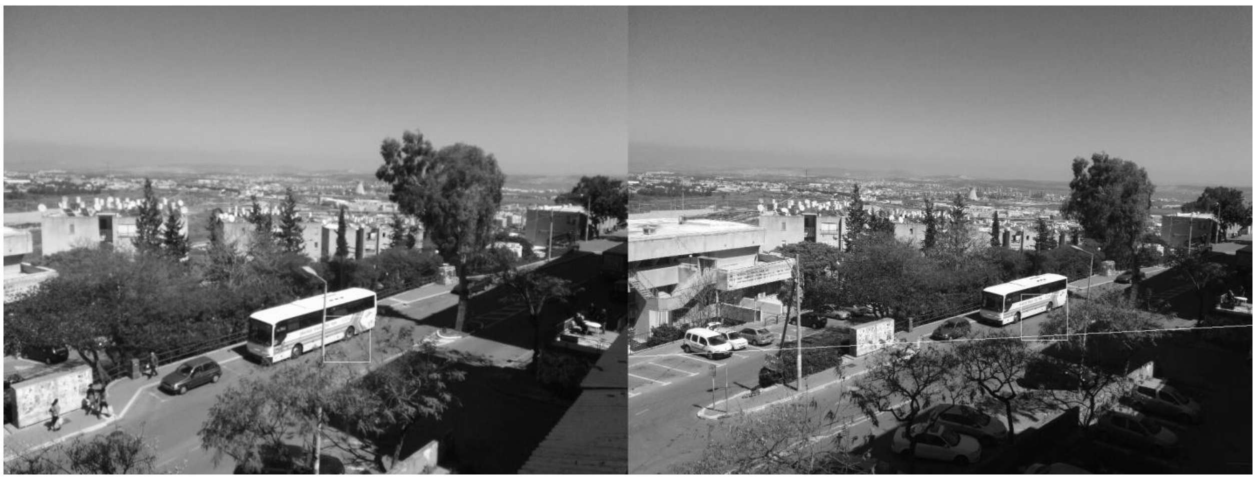
In this case an SSD or hash-base algorithm could help us with identifying the target



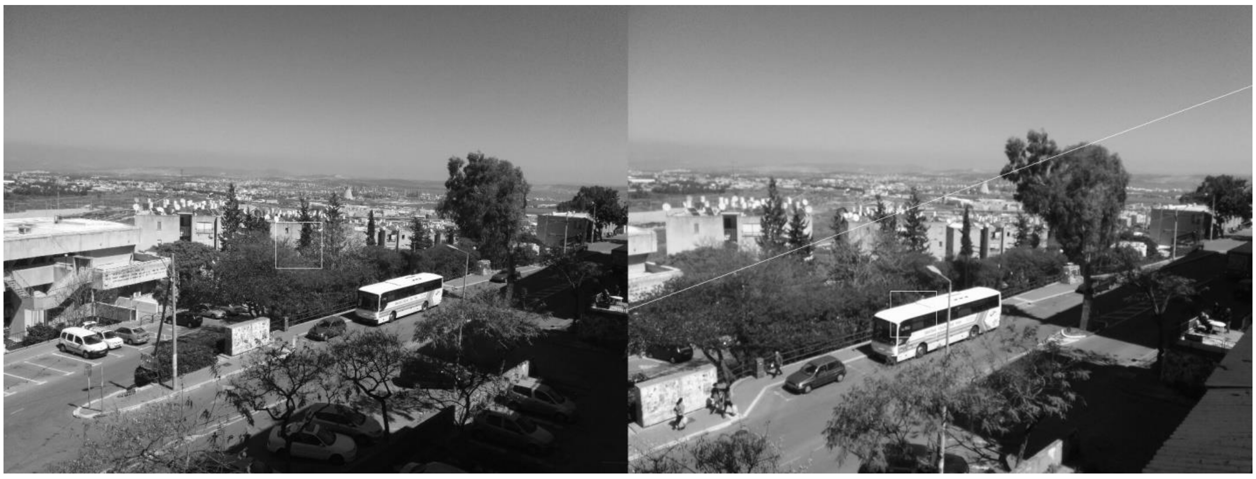
- Homography

Note:of course this approach should not be applied on the whole image because only the same planar surfaces in the space are related by a homography and on the image we have a lot of different surfaces under different orientations, still it was interesting to test it)

Found the correct target



Complete failure



Second test:

- In the second test we used aset of street pictures that contains 5 pictures.
- 10 random (x,y) coordinates were chosen
- Each picture in the set was tested against the other pictures in the set

Success rate which we received

	RANSAC	**SOREPP**
SSD	**30%**	**60%**
aHash	**32.3%**	**51.6%**
pHash	**25.8%**	**56.5%**
Three Points	**22.6%**	**43.5%**
Homography	**32%**	
Competitive	**30%**	**67.7%**

Note: the points which were chosen in the sky or the points which cannot be found on another picture were not counted.

As it could be seen, competitive selection increases overall success rate in case of SOREPP. Thus we can conclude that this algorithm allows more accurate estimation. Following table shows success rate of the Competitive validation, i.e. percentage of correctly chosen winners among all times when at least one algorithm, found a correct point.

	RANSAC	**SOREPP**
Competitive	**61.3%**	**76.4%**

5) Conclusions

(assuming image 1 with target information is sent from device 1 to device 2 which contains image 2):

- if the target point on image 2 is scaled compared to the target point on image 1 then aHash algorithm should give good results (due to its nature)
- if the target point on image 2 is slightly rotated or blurred then SSD or pHash algorithm should be applied
- if the target point on image 2 is occluded by some object then homography or 3points location matching algorithm should be applied
- of course it is very difficult to identify which of the cases we face, that's why competitive approach described in this paper can be applied to identify the best result

6) Future work

a) dHash

While aHash focuses on average values and pHash evaluates frequency patterns, dHash tracks gradients. dHash algorithm works on the difference between adjacent pixels. This identifies the relative gradient direction. Author of the article (link below) suggests that an algorithm performs better than aHash and pHash.

b) AD, CC, NC

Except SSD there are to other similar approaches, each of them has its strengths and weaknesses, for example:

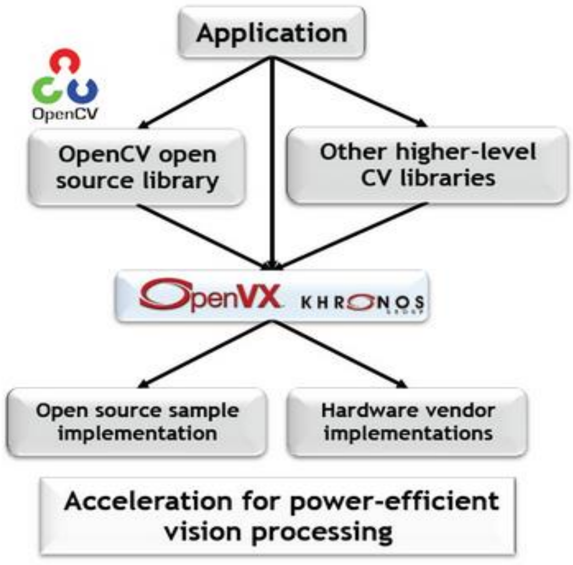
Absolute difference (AD) – AD $= \sum\sum(I_{left} - I_{right})$

Cross correlation (CC) – CC $= \sum\sum(I_{left} * I_{right})$

Normalized correlation (NC) – NC $= \frac{\sum\sum(I_{left} * I_{right})}{\sqrt{\sum\sum(I_{left} * I_{right})}}$

c) OpenMP, OpenCL, OpenVX

- For feature detection and extraction SIFT is one of the best solutions today, but on smartphones it is slow (SIFT processing takes ~15 seconds for images 1280X960 on Nexus5).
- To solve this problem there is new standard currently being developed called OpenVX which allows smartphone vendors to optimize HAL for computer vision algorithms. For multicore smartphones OpenCL and OpenMP may also speed up the processing.


Application
OpenCV
OpenCV open source library
Other higher-level CV libraries
OpenVX KHRONOS
Open source sample implementation
Hardware vendor implementations
Acceleration for power-efficient vision processing

d) Real-time application may be developed for low-resolution pictures (SIFT is much faster on low resolution pictures). Information about the target can be sent from device 1 to device 2. Device 2 will be able not only to find the target but also track it if the device changes its position.

7) External Links

- GeoCam application may be used to get orientation and rotation data: https://play.google.com/store/apps/details?id=com.myway
- Information about pHash algorithm: http://www.phash.org/
- Wi-Fi direct overview information - Device to device communications with Wi-Fi Direct: overview and experimentation from: http://www.campsmur.cat/
- An article from Wikipedia about RANSAC: http://en.wikipedia.org/wiki/RANSAC
- An article about homography: http://en.wikipedia.org/wiki/Homography_(computer_vision)
- An article about aHash and pHash: http://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html
- Good explanation about Gimbal lock: http://www.youtube.com/watch?v=rrUCBOlJdt4
- Feature descriptor comparison report: http://computer-vision-talks.com/articles/2011-08-19-feature-descriptor-comparison-report/