Honda Research Institute Code Sprint

Stereo-based road area detection

Timur Ibadov

1 Introduction

The goal of this code sprint is detection of road surfaces on point clouds, presented by a sequence of images and corresponding disparity maps, with known pose of the camera on each frame. Proposed algorithm is based on [1] and consists of two parts. The first part is a conversion from a disparity map to a Digital Elevation Map (DEM) structure. The remaining part is detection of the road surfaces and two curbs on the DEM.

2 Datasets

Provided dataset consists of 6440 frames. Each frame represented by two rectified images, which were received from a wide-baseline (84cm) stereo-camera on the top of the car, precomputed disparity map and 6DOF relative pose vector in according to the first frame. Also, each frame was manually labeled. See Figure 1 for example of input frame.

3 Point cloud building

If a disparity map and the camera calibration are known, a point cloud of the scene can be computed. $z = focal \cdot baseline/d$, $x = z \cdot (u - u_{center})/focal$, $y = z \cdot (v - v_{center})/focal$, where (x, y, z) — 3D coordinates of a point in the point cloud, (u, v) — coordinates of the corresponding pixel on the left image, d — a disparity of this pixel, and $focal, baseline, u_{center}, v_{center}$ — the intrinsic camera parameters.

4 DEM generation

Digital Elevation Map (DEM) is a horizontal grid, holding a local height value for each grid cell, computed from the triangulated 3D points.
The utilized DEM is defined as a horizontal grid, that is regular and paraxial in the column-disparity space (u, d) as demonstrated in Figure 2. This ensures an approximate constant number of observations assigned to each grid cell, opposed to a grid being regular and paraxial to the horizontal world axes.
We assign all image pixels $(u_k, d_k), k \in \Omega$ of a region of interest Ω that have valid disparity values d_k to their nearest grid cells. From all triangulated height values y_k of the image points assigned to a cell i a common height value y_i is computed using a histogram based approach. For a shorter notation we denote the vector of all height values y_i by $\mathbf{h}$.

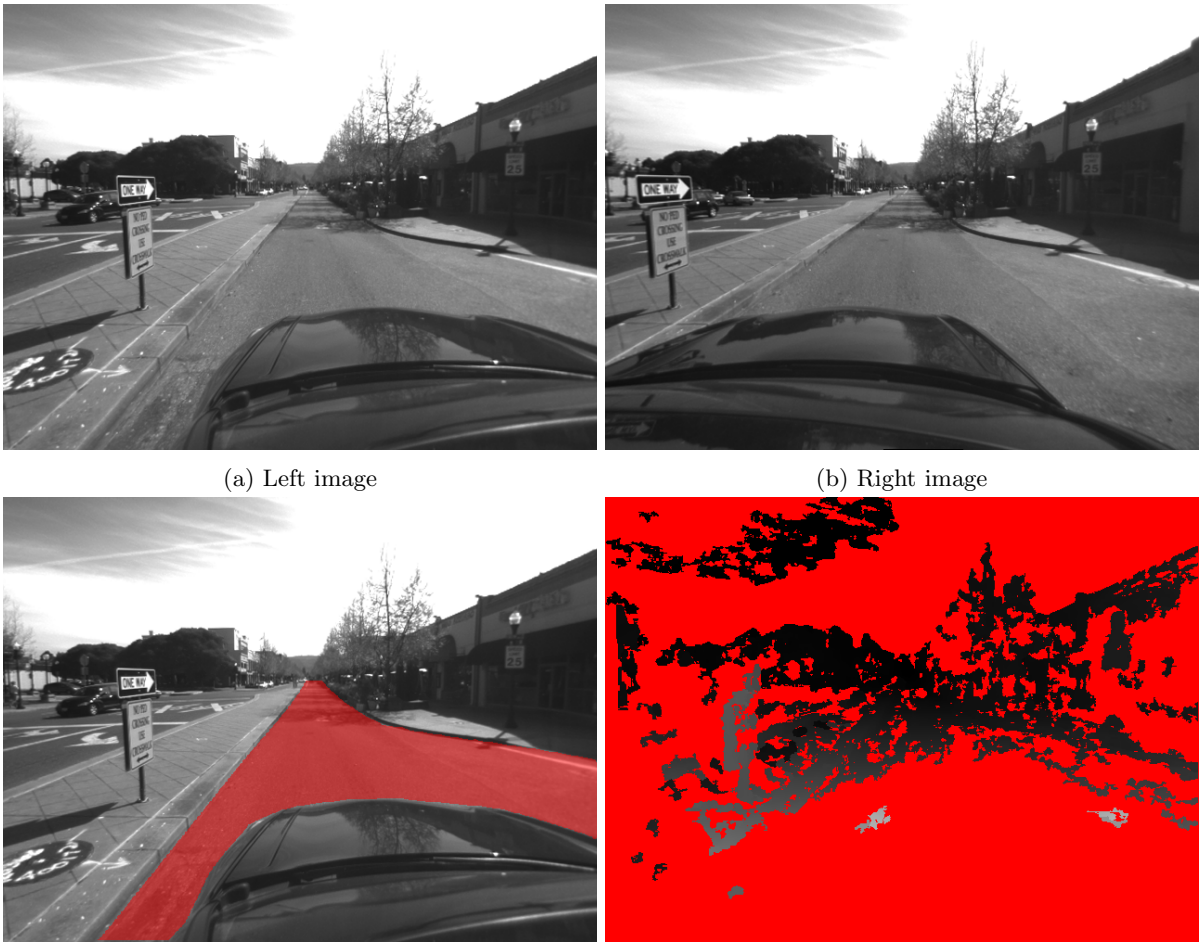
5 Road area detection

It's supposed here, that there is only one curb on the right side of the road. A curb on the left side can be dealt with in a similar way.

5.1 Scene model

Let's define the curb C as a vertical structure, which divide a road surface S and a sidewalk surface A.

$C = \{(x, y, z) \mid x = f_c(z) := [z^3, z^2, z, 1] \cdot \mathbf{c}\}, \quad \mathbf{c} = [c_0, c_1, c_2, c_3]^T \quad .$



(a) Left image (b) Right image

(c) Ground truth (d) Disparity map (red pixels have no data)

Figure 1: A frame from the Castro dataset

Using the vector $\mathbf{q} = [x^2, z^2, 2xz, x, z, 1]^T$, we represent S and A by bounded second order surfaces g_s and g_a.

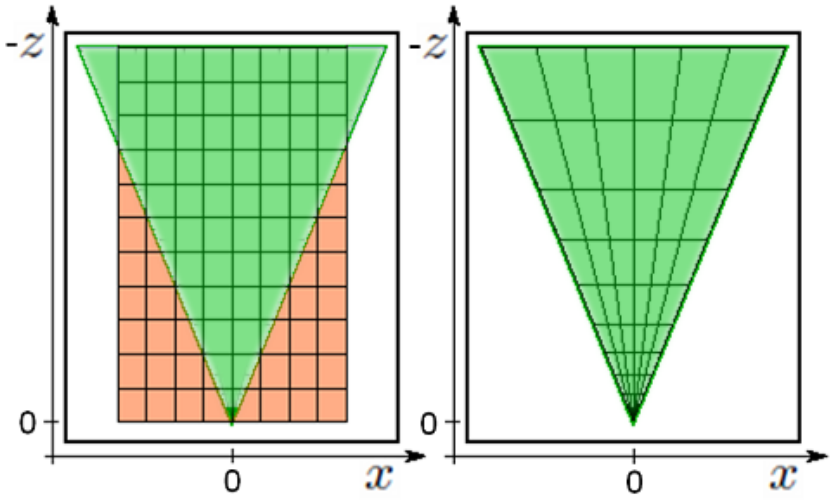
$S = \{(x, y, z) \mid y = g_s(x, z) := \mathbf{s}^T \cdot \mathbf{q}, x \leqslant f_c(z)\}$,
$A = \{(x, y, z) \mid y = g_a(x, z) := \mathbf{a}^T \cdot \mathbf{q}, x \geqslant f_c(z)\}$,

with $\mathbf{s} = [s_0, ..., s_5]^T$, $\mathbf{a} = [a_0, ..., a_5]^T$ being the unknown surface parameters.
We use the notation $\Theta = (\mathbf{c}, \mathbf{s}, \mathbf{a})$ to combine the set of unknown model parameters.

5.2 Classification algorithm

Parameters Θ will be estimated via two-step iterated algorithm (assume a known initial labeling $\boldsymbol{l}_t^{(0)}$):

- Estimation of $\Theta_t^{(n)}$ based on labeling of the previous iteration $\boldsymbol{l}_t^{(n-1)}$.
- Classification $\boldsymbol{l}_t^{(n)} = [{l_{1,t}}^{(n)}, ..., {l_{I,t}}^{(n)}]$ of all DEM cells by labels $l_{i,t}{}_t^{(n)} \in \Lambda = \{\texttt{street}, \texttt{street adjasent}, \texttt{unassigned}\}$ based on $\Theta_t^{(n)}$.



h

Figure 2: Left — DEM in (x, y) space, right — in (u, d) space. Field of view of the camera is marked with green

We will stop iterating, if required accuracy is achieved, or if maximum number of iteration is reached. Labels "`street`" and "`street adjacent`" correspond to surfaces S and A, and label "`unassigned`" corresponds to all other points. $\boldsymbol{l}_t^{(0)}$ is initialized with labeling from the previous frame $\boldsymbol{l}_{t-1}$. In another case left side of DEM is labeled as "`street`", and right one as "`street adjacent`".

5.2.1 Parameter estimation step

Using all DEM cells, labeled as "`street`" we will get an estimation of $\mathbf{s}^{(n)}$:

$$\mathbf{s}^{(n)} = \underset{\mathbf{s}}{\operatorname{argmin}} \left(\sum_{i \in I_s^{(n)}} \frac{1}{\sigma_{h_i}^2} (h_i - g_s(x_i, z_i))^2 \right)$$

Analogously for "`street adjacent`" we will get $\mathbf{a}^{(n)}$:

$$\mathbf{a}^{(n)} = \underset{\mathbf{a}}{\operatorname{argmin}} \left(\sum_{i \in I_a^{(n)}} \frac{1}{\sigma_{h_i}^2} (h_i - g_a(x_i, z_i))^2 \right).$$

Further, the variances σ_s^2 and σ_a^2 of the measured height values with respect to the estimated surfaces are computed.
Next we want to compute curb parameters $\mathbf{c}^{(n)}$. For this we introduce an auxiliary function $g_{b,c}(x, z)$:

$g_{b,c}(x, z) = \frac{2}{1+\exp(b(f_c(z)-x))} - 1.$

Then $\mathbf{c}^{(n)}$ can be computed as:

$$\mathbf{c}^{(n)} = \underset{\mathbf{c}}{\operatorname{argmin}} \left(\sum_{i \in I_s^{(n)} \cup I_a^{(n)}} \omega_i^2 (\phi_i - g_{b,c}(x_i, z_i))^2 \right), \text{ where}$$

$$\phi_i = \begin{cases} -1 & , l_{i,t}^{(n-1)} = \texttt{street} \\ 1 & , l_{i,t}^{(n-1)} = \texttt{street adjacent} \end{cases},$$

$\omega_i = p(l_{i,t}^{(n-1)})$.

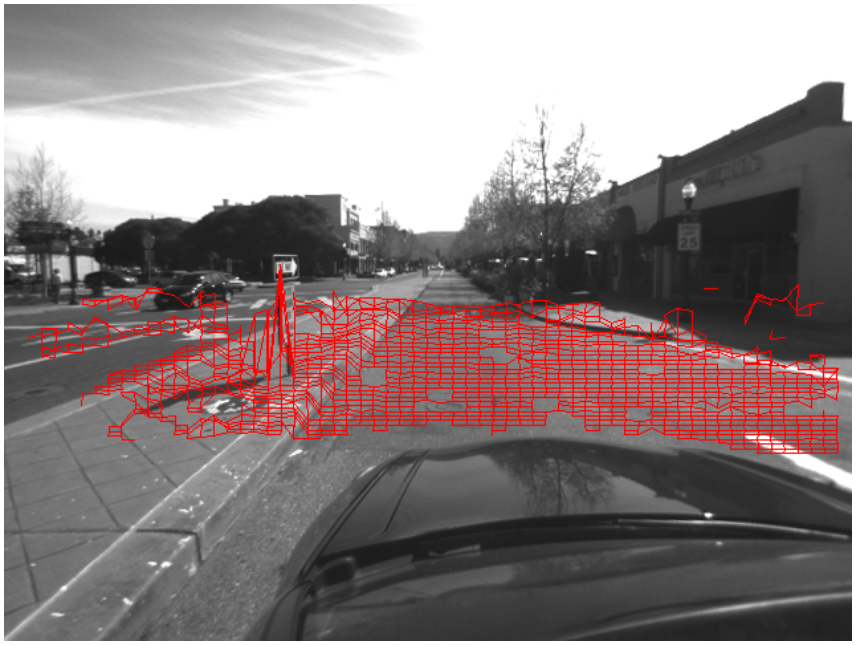


Figure 3: Projection of the DEM to the corresponding left image

5.2.2 Classification step

A labeling is choosing as the most likely labeling considering heights $\mathbf{h}_t$ and $\mathbf{h}_{t-1}$ and scene parameters $\mathbf{\Theta}_t^{(n)}$ and $\mathbf{\Theta}_{t-1}^{(n)}$:

$$\boldsymbol{l}^{(n)} = \underset{\mathbf{l}}{\operatorname{argmax}} \left(p(\boldsymbol{l}_t \mid \mathbf{h}_t, \mathbf{\Theta}_t^{(n)}, \mathbf{h}_{t-1}, \mathbf{\Theta}_{t-1}^{(n)}) \right).$$

But this likelihood can be factorized on production of unary, binary and temporary factors:

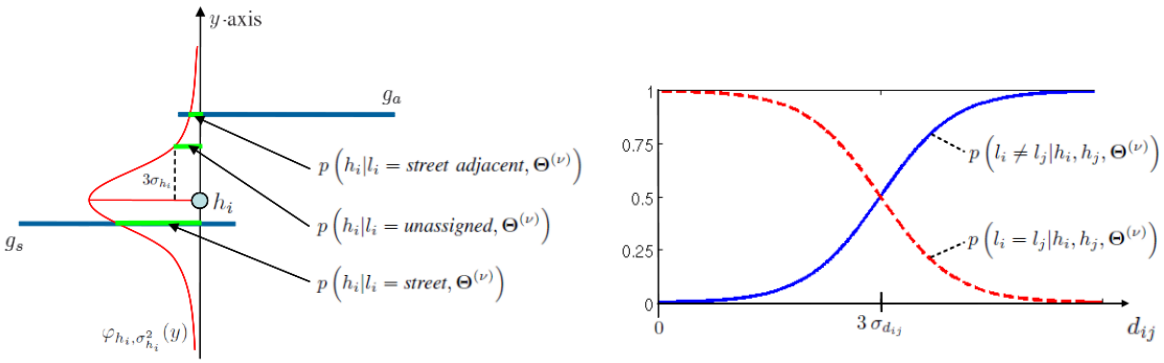
$$p(\boldsymbol{l}_t \mid \mathbf{h}_t, \mathbf{\Theta}_t^{(n)}, \mathbf{h}_{t-1}, \mathbf{\Theta}_{t-1}^{(n)}) \propto \prod_{i \in I} p(l_{i,t} \mid h_{i,t}, \mathbf{\Theta}_t^{(n)}) \prod_{(i,j) \in N_4 \subset I \times I} p(l_{i,t}, l_{j,t} \mid h_{i,t}, h_{j,t}, \mathbf{\Theta}_t^{(n)}) \prod_{i \in I} \Upsilon(l_{i,t} \mid h_{i,t-1}, \Theta_{t-1}).$$

Unary terms.
In according to the Bayes' rule:
$p(l_i \mid h_i, \mathbf{\Theta}^{(n)}) \propto p(h_i \mid l_i, \mathbf{\Theta}^{(n)}) p(l_i \mid \mathbf{\Theta}^{(n)})$

Let's define Gaussian function $\psi_i(y) = \frac{1}{\sigma_{h_i}\sqrt{2\pi}} e^{-\frac{(y-h_i)^2}{\sigma_{h_i}^2}}$, then probabilities will be follows (with ξ as a normalization coefficient):

l_i	$p(h_i \mid l_i, \mathbf{\Theta}^{(n)})$	$p(l_i \mid \mathbf{\Theta}^{(n)})$
"`street`"	$\psi_i(g_s^{(n)}(x_i, z_i))$	$\frac{1}{\xi}\sigma_s^{2(n)} \lVert 1 - g_{b,c}^{(n)}(z_i) \rVert$
"`street adjacent`"	$\psi_i(g_a^{(n)}(x_i, z_i))$	$\frac{1}{\xi}\sigma_s^{2(n)} \lVert 1 - g_{b,c}^{(n)}(z_i) \rVert$
"`unassigned`"	$\psi_i(h_i + 3\sigma_{h_i})$	$\frac{1}{\xi}$

Binary terms.
The binary terms consider the height difference information of neighboring cells. We assume neighboring cells i and j to be more likely assigned with the same labels if the height difference $d_{i,j} = \lVert h_i - h_j \rVert$ is small. Vice versa, we assume them to be labeled different if the height difference is large.



(a) $p(h_i \mid l_i, \boldsymbol{\Theta}^{(n)})$ (b) $p(l_i, l_j \mid h_i, h_j, \boldsymbol{\Theta}^{(n)})$

To distinguish real height differences from measurement noise, the value of $d_{i,j}$ where both options are equiprobable is set to $3\sigma_{d_{i,j}} = 3\sqrt{\sigma^2_{h_i} + \sigma^2_{h_j}}$. The resulting likelihood functions regarding equal and unequal labeling are plotted in Figure 4b.

Temporary terms.
$\Upsilon(l_{i,t} \mid h_{i,t-1}, \Theta_{t-1})$ defined as follows:

- Compute the Cartesian coordinates $(x_{i,t-1}, z_{i,t-1})$ with respect to the previous frame.
- Transform these coordinates into column disparity space $(u_{i,t-1}, d_{i,t-1})$.
- If $(u_{i,t-1}, d_{i,t-1})$ lies inside the previous DEM $\mathcal{M}_{t-1}$:
  - Identify its four nearest neighbor cells $n_1, ...n_4 \in \mathcal{M}_{t-1}$.
  - For all $l \in \Lambda$: Interpolate $\Upsilon(l_{i,t} \mid h_{i,t-1}, \Theta_{t-1})$ from marginals $p(l_{j,t-1} = l \mid \mathbf{h}_{t-1}, \boldsymbol{\Theta}^{(n)}_{t-1})$, where $j = n_1, ..., n_4$.
- Otherwise set $\Upsilon \equiv 1$, i.e. the influence of the prior term for the current cell is switched off.

5.3 Checking the presence of curbs

After iterating we must check, if both curbs are really presented in the scene. For every disparity value along a curb we compute the difference between height y_i from the left and from the right side of the curb. All this differences are accumulated, and if their median value is less than a threshold, we decide, that this curb is not presented in the scene.

6 Results

We use standard precision and recall for evaluating of results. But we consider only pixels with known disparity. So, results are shown on Figure 4.

Mean precision: 89.87%
Mean recall: 52.12%
Mean F-score: 62.45%

Max precision: **100.0%**
Recall: 87.04%
F-score: 93.07%

Max recall: **89.77%**
Precision: 98.90%
F-score: 94.11%

Max F-score: **94.22%**
Precision: 99.97%
Recall: 89.09%

Figure 4: Results on the Castro dataset

7 Contributing to PCL

7.1 DisparityMapConverter

- `stereo/include/pcl/stereo/disparity_map_converter.h`
- `stereo/include/pcl/stereo/disparity_map_converter.hpp`
- `stereo/src/disparity_map_converter.cpp`

Compute point cloud from the disparity map.

7.2 DigitalElevationMapBuilder

- `stereo/include/pcl/stereo/digital_elevation_map.h`
- `stereo/include/pcl/stereo/digital_elevation_map.hpp`
- `stereo/src/digital_elevation_map.cpp`

Build a Digital Elevation Map in the column-disparity space from a disparity map and a color image of the scene.

7.3 DEMGroundSegmentation

- `apps/include/pcl/apps/dem_ground_segmentation.h`
- `apps/src/dem_ground_segmentation.cpp`
- `apps/src/dem_ground_segmentation_example.cpp`

Segment sequences of the Digital Elavation Map for road and sidewalks.

8 Installation and running

First of all you need to install two libraries: NLopt and OpenGM. Instructions, how to do it for your operating system you can find here:
NLopt: `http://ab-initio.mit.edu/wiki/index.php/NLopt#Download_and_installation`.
OpenGM: `http://hci.iwr.uni-heidelberg.de/opengm2/download/opengm-2.0.2-beta-manual.pdf`.
Next you need to compile the experimental PCL release. Instructions are here. Notice, that you should set the following cmake options to true:

- `BUILD_apps`
- `WITH_NLOPT`
- `WITH_OPENGM`

After compiling the library, you will be able to find a binary file `pcl_dem_ground_segmentation_example` in the folder with others PCL binaries.
So, you can run the algorithm now. You need to type:

```
pcl_dem_ground_segmentation_example -l LeftImages -d DisparityMaps -g ResultDir -p pose.dat
```

where:

- `LeftImages` is a folder with left images.

- `DisparityMaps` is a folder with disparity maps.
- `ResultDir` is a resulting folder.
- `pose.dat` is a file with absolute camera positions, relative to the first frame.

You can download a short sequence of frames from the Castro dataset from here.
After computation will be finished, in the `ResultDir` you will find a labeling file for every frame and two folders, `Left` and `Right`, that contain curbs coordinates on each frame.
Some MATLAB scripts, which can be helpful to visualization of that result you may find at Appendix A.

9 References

[1] Siegemund, J., Franke, U., & Forstner, W. (2011, June). A temporal filter approach for detection and reconstruction of curbs and road surfaces based on conditional random fields. In Intelligent Vehicles Symposium (IV), 2011 IEEE (pp. 637-642). IEEE.

[2] Siegemund, J., Pfeiffer, D., Franke, U., & Forstner, W. (2010, June). Curb reconstruction using Conditional Random Fields. In Intelligent Vehicles Symposium (pp. 203-210).

A MATLAB's scripts

A.1 Generate a video from results

Generating a video `res_video.avi` and a set of images with labeling in a folder `GeneratedImageDir` from the results. Usage:
`save_video_from_labels(LeftImages, ResultDir, GeneratedImageDir, 'res_video.avi',28, 50);`

```
function save_video_from_labels (imageDir, labelsDir, ...
            resultImageDir, videoFileName, frameRate, quality)

% Create AVI object.
vidObj = VideoWriter(videoFileName);
vidObj.Quality = quality;
vidObj.FrameRate = frameRate;
open(vidObj);

% Create movie.
imageFileNames = dir([imageDir '/*.png']);
for i = 1 : length(imageFileNames)
    imageFileName = imageFileNames(i).name;
    dotIndex = strfind(imageFileName, '.');
    dotIndex = dotIndex(end) - 1;
    imageFileName = imageFileName(1:dotIndex);
    disp(imageFileName);
    img = imread([imageDir '/' imageFileName '.png']);
    labels = dlmread([labelsDir '/' imageFileName '.txt']);
    img(labels == 1) = img(labels == 1) / 2;
    colorImage(:, :, 2) = img;
    colorImage(:, :, 3) = img;
    img(labels == 1) = img(labels == 1) + 127;
    colorImage(:, :, 1) = img;

    f = figure('visible', 'off');
    imshow(colorImage, 'Border', 'tight');
    hold on;

    try
        left = dlmread([labelsDir '/Left/' imageFileName '.txt']);
        plot(left(:, 1), left(:, 2), 'LineWidth', 4, 'Color', 'green');
    catch exeption
        % Nothing to do.
    end

    try
        right = dlmread([labelsDir '/Right/' imageFileName '.txt']);
        plot(right(:, 1), right(:, 2), 'LineWidth', 4, 'Color', 'blue');
    catch exeption
        % Nothing to do.
    end

    set(gcf, 'PaperUnits', 'inches', 'PaperPosition', ...
        [0 0 size(img, 2)-1 size(img, 1)]/80);
```

```
    print(gcf, '-dpng', '-r80', ...
        [resultImageDir '/' imageFileName '.png']);
    close(f);
    img = imread([resultImageDir '/' imageFileName '.png']);
    writeVideo(vidObj, img);
end

% Save as AVI file
close(vidObj);

end
```

A.2 Test results

Testing the results with ground truth and saving scores in `result.txt`. Usage:
`test_labeling(ResultDir, GroundTruthDir, DisparityMaps, result.txt);`

```
function [ precision, recall, f_score ] = test_labeling( ...
        labeling_path, gt_path, disparity_path, result_filename )

    if (nargin > 3)
        disparity_filenames = dir([disparity_path '/*.txt']);
    end

    labeling_filenames = dir([labeling_path '/*.txt']);
    gt_filenames = dir([gt_path '/*.txt']);

    number_of_files = min(length(labeling_filenames), length(gt_filenames));

    precisions = zeros(number_of_files, 1);
    recalls = zeros(number_of_files, 1);
    top_x_interest = min(10, number_of_files);

    for label_idx = 1 : number_of_files
        disp(labeling_filenames(label_idx).name);

        labeling = dlmread([labeling_path '/' ...
                labeling_filenames(label_idx).name]);
        gt = dlmread([gt_path '/' gt_filenames(label_idx).name]);

        if (nargin > 3)
            fid = fopen([disparity_path '/' ...
                disparity_filenames(label_idx).name]);
            disparity = fscanf(fid, '%f', inf);
            fclose(fid);

            disparity = reshape (disparity, ...
                    size(labeling, 2), size(labeling, 1));
            disparity = disparity';

            labeling(disparity == -1) = 0;
            gt(disparity == -1) = 0;
```

```
        end

        labeling = labeling == 1;
        gt = gt == 1;

        tp = sum(sum(gt == 1 & labeling == 1));
        fn = sum(sum(gt == 1 & labeling == 0));
        fp = sum(sum(gt == 0 & labeling == 1));
        if (tp ~= 0)
            precisions(label_idx) = tp / (tp + fp);
            recalls(label_idx) = tp / (tp + fn);
        end
    end

    f_scores = 2 .* precisions .* recalls ./ (precisions + recalls);
    f_scores(isnan(f_scores)) = 0;

    precision = mean(precisions);
    recall = mean(recalls);
    f_score = mean(f_scores);

    [~, precisions_idx] = sort(precisions);
    [~, recalls_idx] = sort(recalls);
    [~, f_scores_idx] = sort(f_scores);

    fid = fopen(result_filename, 'wt');

    fprintf(fid,'Tested %d frames.\n\n', number_of_files);

    fprintf(fid,'Mean precision: %.6f\n', precision);
    fprintf(fid,'Mean recall:    %.6f\n', recall);
    fprintf(fid,'Mean F-score:   %.6f\n\n', f_score);

    fprintf(fid,'*** The best results **********\n\n');

    fprintf(fid,'Max precision:\n\n');

    fprintf(fid,'        Filename          | PRECISION |   Recall  |  F-score\n');
    fprintf(fid,'-------------------------------------------------------------\n');
    for label_idx = number_of_files : -1 : number_of_files - top_x_interest
        index = precisions_idx(label_idx);
        fprintf(fid,'%-25s | %8.6f  | %8.6f  | %8.6f\n', ...
            labeling_filenames(index).name, ...
            precisions(index), recalls(index), f_scores(index));
    end
    fprintf(fid,'\n');

    fprintf(fid,'Max recall:\n\n');

    fprintf(fid,'        Filename          | Precision |  RECALL   |  F-score\n');
    fprintf(fid,'-------------------------------------------------------------\n');
```

```
    for label_idx = number_of_files : -1 : number_of_files - top_x_interest
        index = recalls_idx(label_idx);
        fprintf(fid,'%-25s| %8.6f  | %8.6f  | %8.6f\n', ...
            labeling_filenames(index).name, ...
            precisions(index), recalls(index), f_scores(index));
    end
    fprintf(fid,'\n');

    fprintf(fid,'Max F-score:\n\n');

    fprintf(fid,'        Filename          | Precision |   Recall   |  F-SCORE\n');
    fprintf(fid,'-------------------------------------------------------------------\n');
    for label_idx = number_of_files : -1 : number_of_files - top_x_interest
        index = f_scores_idx(label_idx);
        fprintf(fid,'%-25s| %8.6f  | %8.6f  | %8.6f\n', ...
            labeling_filenames(index).name, ...
            precisions(index), recalls(index), f_scores(index));
    end
    fprintf(fid,'\n');

    fprintf(fid,'*** The worst results **********\n\n');

    fprintf(fid,'Min precision:\n\n');

    fprintf(fid,'        Filename          | PRECISION |   Recall   |  F-score\n');
    fprintf(fid,'-------------------------------------------------------------------\n');
    for label_idx = 1 : top_x_interest
        index = precisions_idx(label_idx);
        fprintf(fid,'%-25s| %8.6f  | %8.6f  | %8.6f\n', ...
            labeling_filenames(index).name, ...
            precisions(index), recalls(index), f_scores(index));
    end
    fprintf(fid,'\n');

    fprintf(fid,'Min recall:\n\n');

    fprintf(fid,'        Filename          | Precision |  RECALL    |  F-score\n');
    fprintf(fid,'-------------------------------------------------------------------\n');
    for label_idx = 1 : top_x_interest
        index = recalls_idx(label_idx);
        fprintf(fid,'%-25s| %8.6f  | %8.6f  | %8.6f\n', ...
            labeling_filenames(index).name, ...
            precisions(index), recalls(index), f_scores(index));
    end
    fprintf(fid,'\n');

    fprintf(fid,'Min F-score:\n\n');

    fprintf(fid,'        Filename          | Precision |   Recall   |  F-SCORE\n');
    fprintf(fid,'-------------------------------------------------------------------\n');
    for label_idx = 1 : top_x_interest
```

```
138         index = f_scores_idx(label_idx);
139         fprintf(fid,'%-25s | %8.6f   | %8.6f   | %8.6f\n', ...
140             labeling_filenames(index).name, ...
141             precisions(index), recalls(index), f_scores(index));
142     end
143 
144     fclose(fid);
145 end
```