Unmanned Ground Vehicle, Road Tracking, Comparison With Model.

Vojtech ŠIMÁK[1]

COMPUTER VISION BASED UGV STEERING

Abstract

This report deals with a computer vision system for unmanned ground vehicle. The concentrated aspects include the comparison between image and model and intrinsic and extrinsing parameters of camera.

1. LIST OF MAIN SYMBOLS

UGV	Unmanned Ground Vehicle
GPS	Global Positioning System
RANSAC	RANdom SAmple Consensus
DCT	Discreet Cosine Transformation
IDCT	Inverse Discreet Cosine Transformation
JPEG	Joint Photographic Experts Group
MPEG	Movie Picture Expert Group
DARPA	The Defense Advanced Research Projects Agency
SIFT	Scale Invariant Feature Tracking (Transform)
CPU	Central Processing Unit
GPU	Graphics Processing Unit
CUDA	Compute Unified Device Architecture

2. INTRODUCTION

The Main purposes of UGV are to drive vehicles in harmful environment or to take the monotonous tasks from human. These vehicles are used in military and in industry. The Industrial use of UGV development is in automotive industry in various driver assistance systems.

Computer vision based on navigation is often used in UGV's sensor system [[4]] together with other sensors. Joining this all information in UGV control unit provides simplified model of UGV surrounding. Image analyzing can be used for various purposes e.g. like road tracking, obstacle detection, information about traffic situation, reading registration numbers of vehicles, reading road signs… In this report are described computer vision algorithms used only for road

[1] Vojtech Šimák, MSc. University of Žilina, Faculty of mechanical engineering, Department of industrial engineering

tracking in forest environment, but after some small changes they can be also used in urban environment as well.

3. SYSTEMS STRUCTURE

Computer vision is only one subsystem from systems controlling UGV. In this case it is used only for road tracking and not for obstacle detection.

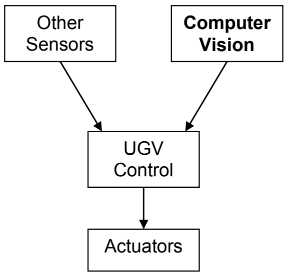


Figure 1 Systems structure

Structure of computer vision is shown in
Figure **2** after saving the image to computer memory (image acquisition), are several different approaches available to analyze this image.

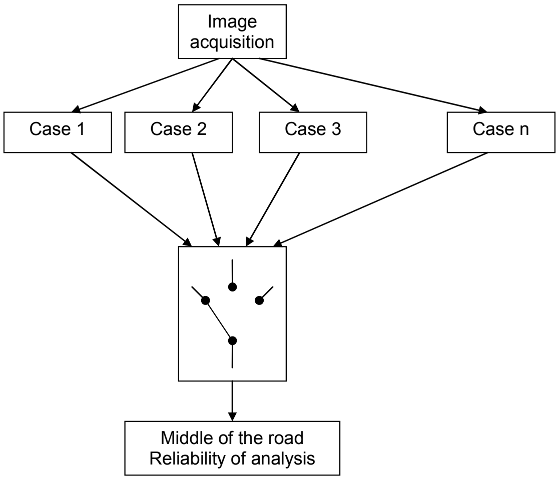


Figure 2 Computer vision structure

4. RESEARCH IN MATLAB ENVIROMENT

In this chapter are described different algorithms used by searching of analysis method. Matlab with image processing toolbox provides very quick and simple tool for various experiments in image processing.

4.1. Case 1

This approach is a result of various experiments of using image filters to extract information about road coordinates. Conditions for successful analysis are:

- Ambient scene illumination (without significant shadows).
- Road has the same color along it from camera view.
- Used in scene with snow outside the road and freeze mud on the road.

Description of the analysis:

- Loading image from file.
- Identification of image resolution.
- Cutting of not important parts of image (engine bonnet and scenery above horizon).
- Tresholding image at 51% (can be calculated automatically, see the last step).
- Sobel filter to increase the vertical lines.
- Canny detector - edge detection.
- Hough transformation with rho resolution 3.9% of image width and theta resolution 10°.
- Detecting peaks of transformation at level 30% of maximum transformation value.
- Detecting lines in image with minimal length of 14,.6% of image width and possible gap of 4,.88% of image width .
- Selecting two lines with the nearest slope to vertical line (high probably the road borderlines).
- Creating of trapezoid from this these two lines.
- Assigning all image points outside the trapezoid (road) to value 0.
- Computing histogram of the image.
- Assigning count of 0 value to 0.
- Integration of the histogram (sum).
- Computing value of 95% of image points.
- Possibility of using this value as tresholding value to next-coming image (for the same image is this value convergent to 51% from both sides in interval circa from 45% to 55%).

4.2. Conclusion to Case 1:

The analysis is loosing road in curves because images taken by camera with greater focal length lens are not scanning the road on inner side of the curve. This problem was solved by replacing old lenses with lenses with shorter focal length.

1. Sometimes is the analysis finding wrong borderline for example between trees in the forest.

4.3. Case 2

In this case I have analyzed the scene without snow (most of year).
The conditions for successful analysis are:

- Ambient scene illumination (without any significant shadows).
- Road is lighter than surrounding.

Description of the analysis:

- Loading image from file.
- Identification of image resolution.
- Cutting of not important parts of image (engine bonnet and scenery above horizon).
- Selecting 5 rows of the image in 1/6, 2/6, 3/6, 4/6 and 5/6 of image height.
- Comparing these rows with ideal road width using cross-covariance.
- Finding maxima of the cross-covariance and projecting it to the image width.
- Rotating coordinate system of selected points about 90 degrees because during approximation y=f(x) is vertical line hard to approximate.
- Polynomial approximation of selected points.
- Rotating coordinate system of selected points and approximation line back about -90 degrees.
- Plotting coordinates of road middle and selected points to the image.

4.4. Conclusion to Case2:

This approach is more reliable than in Case 1, in most cases it is successful. The cross-covariance is the solution for cases when the surface is not Lambertian (wet mud and plashy ground) and tresholding like in Case1 will give wrong results. In some cases this approach fails but the cases are very difficult to analyze (for example plash parallel to the road caused by heavy vehicle) and in cases where is near to no color difference between road and forest.

4.5. Case 3

In this case I have analyzed the scene with snow like in Case1.
Conditions for successful analysis are:

- Ambient scene illumination is not necessary.
- Road is darker than surrounding.

Description of the analysis:

- Loading image from file.
- Identification of image resolution.
- Cutting of not important parts of image (engine bonnet and scenery above horizon).
- Comparing every row between 1/6 and 5/6 of image height with ideal road width using cross-covariance.
- Finding maxima of the cross-covariance and projecting it to the image width.
- Using RANSAC approximation for finding the middle of the road.
- Computing reliability of analysis from result of RANSAC as number of points lying on the line divided by number of all points.
- Plotting the result to output image.

4.6. Conclusion to Case3:

This case is the most reliable and the most robust of previous ones. It was used to analyze images taken by sunlight (the most difficult scene). In some cases it fails, especially when projected shades are not parallel to x or y axis. The RANSAC algorithm makes this analysis more reliable because it throws out wrong solutions of approximation.

4.7. Cross-covariance

Result of cross-covariance between image row and “ideal road width impulse” when n is image width and also the number of impulse elements is 2*n*-1.

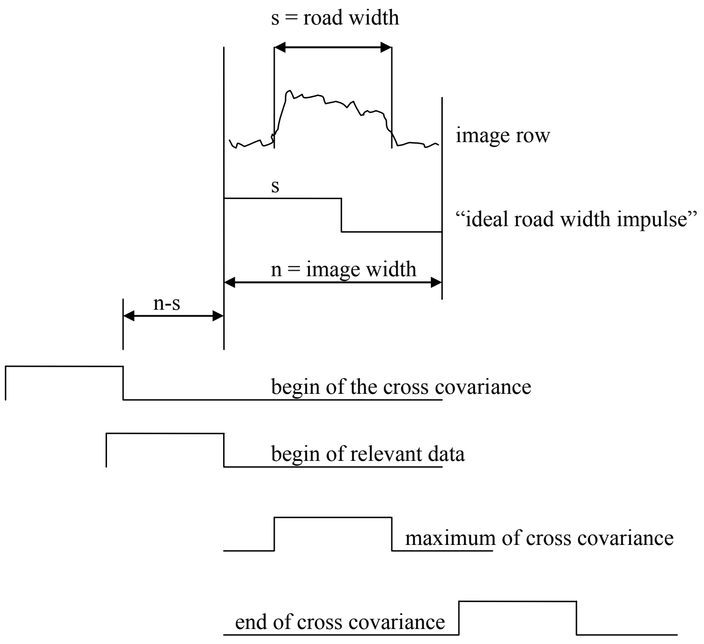


Figure 3 Cross covariance principle

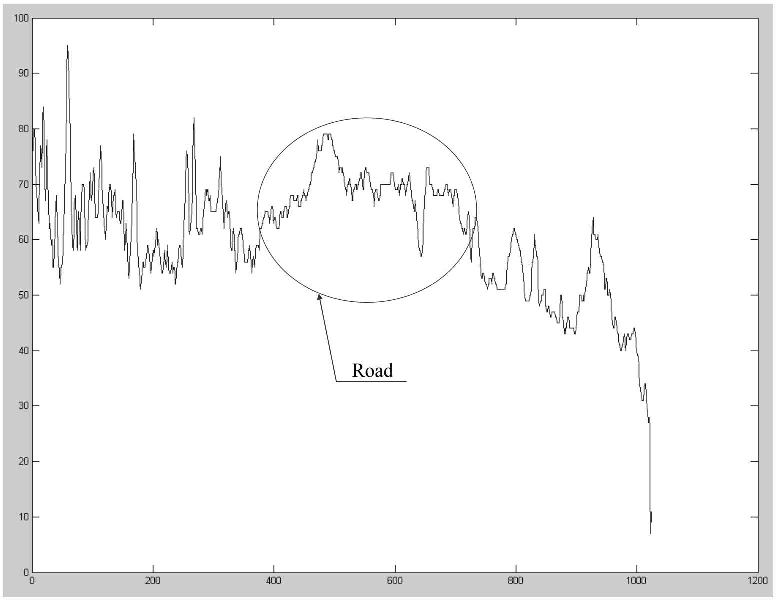


Figure 4 Image row

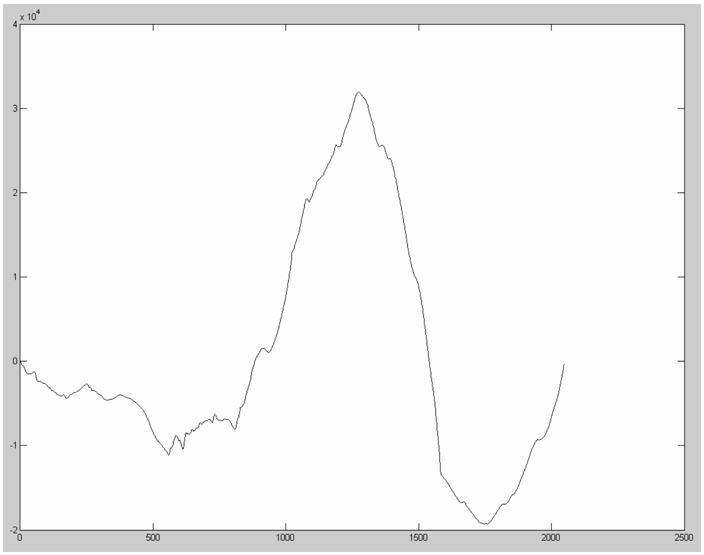


Figure 5 Cross covariance result

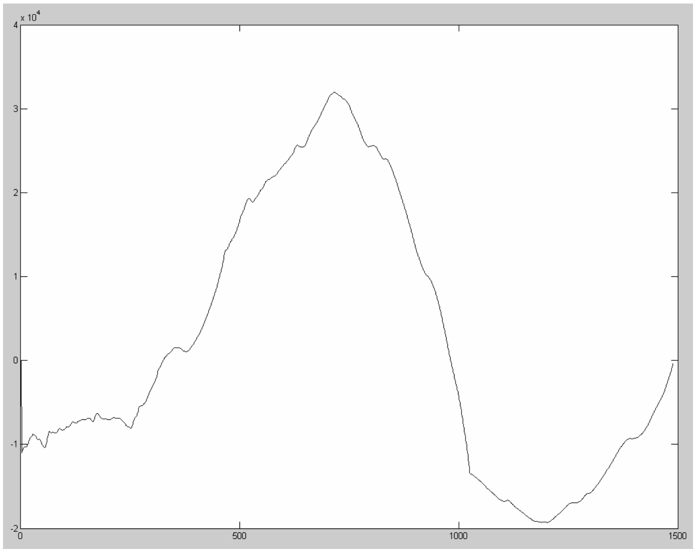


Figure 6 Relevant part of cross covariance result

4.8. Cases with poorer results

- **Using of DCT**. One of the experiments was using of discreet cosine transformation to texture analysis in 32 by 32 blocks. This was not very useful because:
  1. The image is saved as JPEG and all high frequency components are distorted.
  2. The scene taken by a camera is perspective and all objects near to the camera are looking bigger than far away ones (the same objects are low frequency near the car and high frequency when they are farther away).
  3. Removing some components and following IDCT is causing not desired color differences between the blocks.

5. C++ PROGRAM IMPLEMENTATION

This chapter describes implementing of previous research in C++ because it is programming language used by programming of UGV navigation system and code written in C++ is closer to optimal solution than code transferred from Matlab to C++ (also it is not recommended). The second reason to C++ implementation is the similarity between C++ and OpenGL Shading Language (almost the same) and availability of OpenCV library. This chapter also deals with some simplifications and improvements of used algorithms.

5.1. Making cross covariance faster[*]

Standard computing of cross covariance (shown in (1) or (2)) see [3] is processing full impulse length for each shift, but the most values are not changed between next coming shifts (like in
Figure 7).

$$\operatorname{cov}(X,Y) = E\left((X - E(X)).(Y - E(Y))^T\right) \quad where \quad X = (X_1, X_2, \ldots, X_n) \quad and \quad Y = (Y_1, Y_2, \ldots, Y_n)\cdot \qquad (1)$$

or:

$$\operatorname{cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i - E(X)).(Y_i - E(Y)) \text{ where } E(X) = \frac{1}{n}\sum_{i=1}^{n} X_i \quad and \quad E(Y) = \frac{1}{n}\sum_{i=1}^{n} Y_i \qquad (2)$$

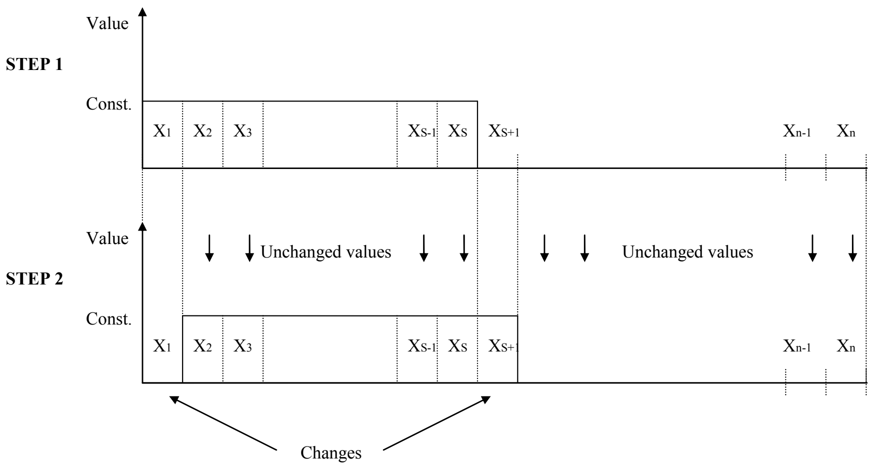


Figure 7 Difference between covariance steps

Step 1. The first computing of cross covariance

$$\operatorname{cov}_1(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i - E(X)).(Y_i - E(Y))\cdot \qquad (3)$$

[*] It means for this case only (not generally) when one of the vectors X or Y are of special shape (impulse with constant value first s elements and value 0 other elements)

Next steps are using first or previous values

$$\begin{aligned}
&\operatorname{cov}_{m+1}(X,Y)=\operatorname{cov}_m(X,Y)\\
&-((const.-E(X)).(Y_m-E(Y)))-((0-E(X)).(Y_{m+s}-E(Y)))\\
&+((0-E(X)).(Y_m-E(Y)))+((const.-E(X)).(Y_{m+s}-E(Y)))\cdot
\end{aligned} \tag{4}$$

$$where \quad m=1,2,\ldots,n-s$$

5.2. Saving of computing power example

When n=1024 and s=350 then number of impulse shifts is $n-s-1$=1024 – 350 – 1=673 during each shift is need to compute the whole comparison (2 x subtraction, 1 x multiplication and 1 x summation) 1024 times, then resulting count of operations is:

2.n.($n-s-1$) = 2 . 1024 . 673 = 1 378 304 times subtraction, n.($n-s-1$) = 689 152 times multiplication and n.($n-s-1$) = 689 152 times summation for each image row.

Using this faster approach is a number of impulse shifts is: $n-s-1$=1024 – 350 – 1 = 673

During each shift is needed to compute only difference between the last comparison and new impulse location (using step = 1 is difference in only two values – begin and end of impulse duration). The First is necessary to compute the whole impulse comparison: 2.n=2048 times subtraction and n=1024 times multiplication and n=1024 times summation. Then it is possible to use the first covariance result in the next coming steps and then to calculate only the difference between these impulses: 10 x subtraction, 4 x multiplication and 2 x summation, then the resulting count of operations is: 10.($n-s-1$) + 2n= 8778 times subtraction, 4.($n-s-1$) + n= 3716 times multiplication and 2.($n-s-1$) + n= 2370 times summation.

In the table:

Tab. 1 Saving of computing power

Operations	Universal approach	Simplified approach	Saving of time [%]
Subtraction	1 378 304	8778	99,363
Multiplication	689 152	3716	99,461
Summation	689 152	2370	99,657

However, this simplified approach is possible <u>only</u> in case when one of the impulses is of <u>special shape</u> described above!

When it is no need to know the accurate covariance result, only the position of maxima or its shape, then we can assign the first covariance result to arbitrary value (for example 0). Then it is possible to save additional time (related to the simplified approach) it is no need to compute the first covariance value with 2n times subtraction, n times multiplication and n times summation.

$$when \quad \operatorname{cov}_1(X,Y) \quad is \quad assigned \quad to \quad 0 \quad then:$$

It is need to compute only 10.($n-s-1$)=6730 times subtraction, 4.($n-s-1$)=2692 times multiplication and 2.($n-s-1$)=1346 times summation.

Tab. 2 Additional time saving

Operations	Simplified approach	Simplified approach 2	Saving of time [%]
Subtraction	8778	6730	23,331
Multiplication	3716	2692	27,557
Summation	2370	1346	43,207

Resulting analysis time on computer AMD Sempron 3000+, 1.81 GHz with 448 MB RAM and operating system Windows XP is for image resolution 512 x 384 (analyzing 100 rows) 16 ms and for image resolution 1024 x 768 (analyzing 200 rows) 120 ms.

5.3. Processing the covariance result

Can be solved with 2 algorithms based on RANSAC [[10]].

Improved RANSAC

- Choose 2 random points (p_m,p_n) from covariance results
- Compute line parameters (x_{0j},$tg(\alpha_j)$) for line l_j joining these points ($l_j; p_m, p_n \in l_j$)
- In defined surrounding of this line search for other covariance results $p_i \in l_j \pm \varepsilon$
- Compute "weight" of the line $w_j = \sum_{\forall p_i \in l_j \pm \varepsilon} \frac{r.dist_i + s}{r + s}$ where:

N is number of covariance results
$dist_i$ is distance between line l_j and covariance result p_i (from 0 to 1)
0 is far away (ε), 1 is on the line
r distance weight coefficient
s solution weight coefficient

- Repeat it M times
- Accept solution with maximum weight $solution(x_{0j},tg(\alpha_j)) = max(w_j)$ for $j=0,1,\ldots,M$ of the line l_j

Fitting the most probably solution

- Choose 2 random points from covariance results
- Compute line parameters joining these points and write it to x_{0j},$tg(\alpha_j)$ (similar to Hough transformation)
- Repeat it M times ($j=0,1,\ldots,M$)
- Sort array $tg(\alpha)$ from the smallest value to the largest ones
- Search for cluster with smallest value dispersion
- Middle of this cluster in $tg(\alpha)$ array is the most probably line direction (see
- Figure **8**)
- For each solution in x_0 array search for points in line surrounding like in previous approach but only for the direction computed in previous step
- Line with most points in the surrounding is the most probably solution

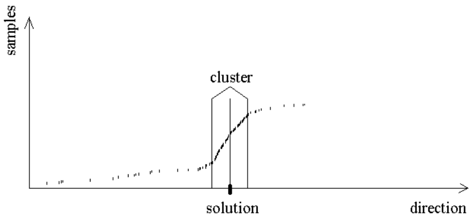


Figure 8 Searching for cluster in tg(α) array

5.4. Program output

To provide a good connectivity to other navigation systems it is need to compute information about:

- Side deviation between the vehicle reference point (projected onto the road) and the middle of the road
- Angular deviation between longitudinal vehicle axis and road axis
- Confidence of the analysis (value between 0 and 1)

5.5. Measuring of camera extrinsic parameters

Main idea was to measure the height of camera (1,53m ± 0,005m). Then to put a white tape on a wooden rod with lower border 1,53m from end and to take images from 10 and 20 meter distance (see
Figure **9**).

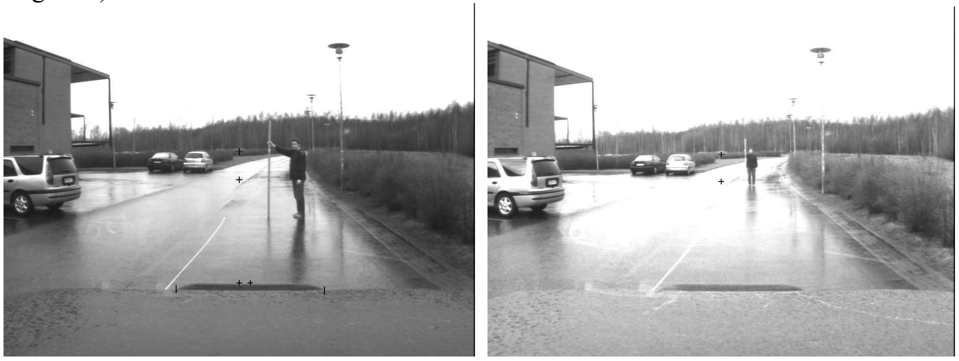


Figure 9 Measuring of camera extrinsic parameters

The upper cross mark is in the same high as camera (on horizon) and the lower in the middle of the image. Difference in image is 60±1 pixels and in 10±0,05 m distance is it

646±10,77 mm (considering ideal camera parameters) camera angle downwards is then 3° 41'±4'.

The frontal bumper is mounted in the middle of the vehicle; one cross mark is in the middle of the bumper and another one in the middle of this image row. The difference between cross marks is 24±1 pixels and the camera angle rightwards is then 1° 25'±4'. Angular and distance error is related to ±1 pixel error that can occur.

5.6. Algorithm for UGV steering

After measuring of camera extrinsic parameters (see previous chapter) is possible to use this information for car control. Before sending command to steering servo is necessary to filter it using following steps:

- **Determining of maximum change between frames**. This step filters signal changes higher than saturation value. In this UGV is command accepted by steering computer from –100% (left end position of steering wheel) to +100% (right end position). This change from –100% to +100% takes about 5 seconds (maximum servo speed limitation). Between following frames (0,2 s) is maximal acceptable command ± 8%. This saturation value is set to ± 5%.
- **Determining of left and right end positions**. By car driving are people turning their head or eyes to the curve. In this UGV is camera rigid to the UGV body and by maximum curvature (by –100% or +100%) is the turning diameter only 12,009 m [9]. The main sensing field is from 4,55m to 24,57m from reference point of UGV so that near the whole curve is outside the camera sensing field. In this case are the end positions set to ± 50% (cca 20 m curve diameter).
- **Steering delay**. By UGV steering is necessary to send commands about actual vehicle - road position, because searching algorithm is scanning area from 4,55m in front of reference to 24,57m from reference, and curve is approximated by line. In
- Figure **10** is shown the reason for steering delay. By instant change of front wheels angle to corresponding image will UGV steer outside the road (to inner borderline). So it is need to wait till the projected middle of the road is near enough to reference point. To smooth this change is now the vehicle turning also before this reference – road intersection. For example 3m from intersection it is turning only with 1/3 command, 2m with 1/2 command and 1m from intersection or behind it with full command.

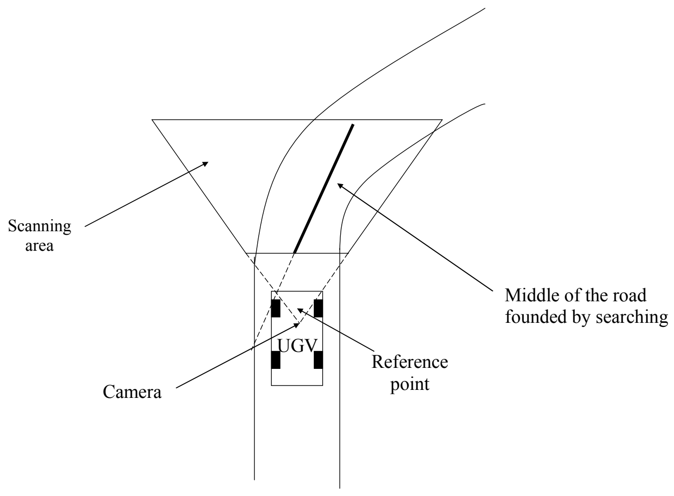


Figure 10 Processing road-vehicle position

- **Image error**. By finding road outside the scanning area is used previous result for this steering calculation.

5.7. Scale Invariant Feature Tracking (SIFT)

This approach looks very promising for this case of computer vision. It can be used not only for road tracking, but also for obstacle detection.

This approach was tried out by Sami Terho (processing road images using his software). Unfortunately without any simplifying it is not real time analysis using CPU (processing time between images in seconds) but it can be made real time (see next chapter)

- Selecting features in image *N* (for example using Harris filter)
- Finding corresponding features in image *N*+1
- Computing motion vectors between features in image *N* and *N*+1
- Computing essential matrix-*E* (for example using RANSAC and 8-point algorithm) and deleting wrong motion vectors (not corresponding to *E* in most cases)
- 3D space reconstruction of surroundings.

5.8. Using GPU to image analysis

Using graphics card supporting OpenGL Shading Language can be very promising to make a difficult image processing (filtering, transformations…) real time. For example graphics card GeForce 8800 GTX from NVidia has a computing power of 345.6 GFLOP/s [8]. The architecture of GPU is massive parallel and it is necessary to distribute the task to for example 128 processing units, however in most cases it is not difficult, but in some special cases it is necessary to compute with whole image matrix.

5.9. Switch process

To find a universal method of computer vision or road tracking it is near to impossible for this time computers. When we are using some several different approaches we need a criterion to choose one or another approach from this set. Sometimes it is possible to take the reliability of analysis and to choose the approach with the highest reliability. In this developing stage it is needed to choose the approach manually.

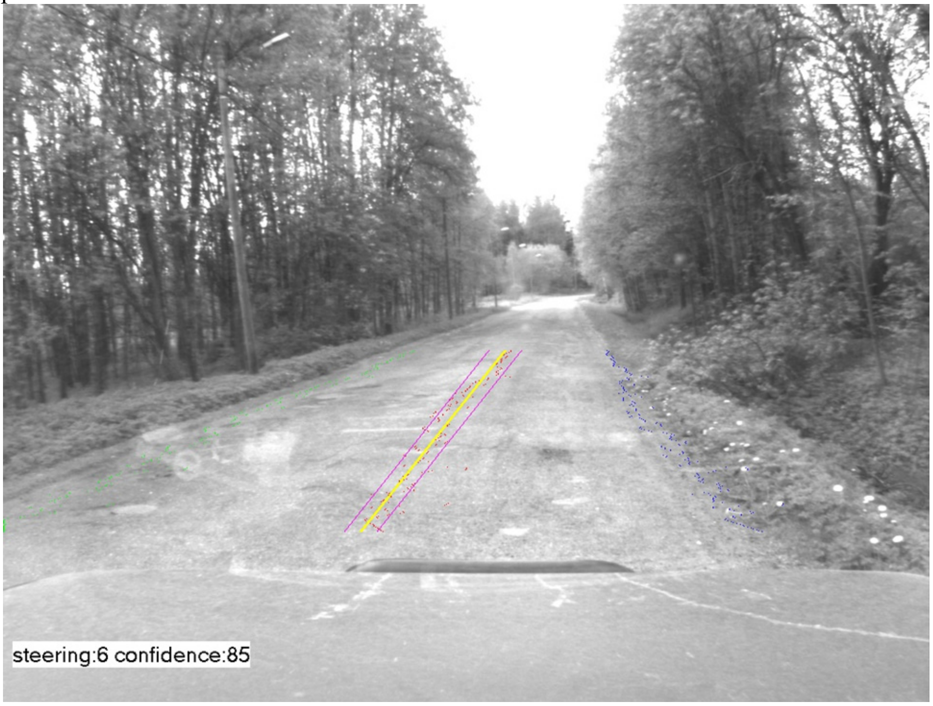
5.10. Reference point of the vehicle

Reference point is in the middle of the front axle at height of lowest part of the body (ground clearance) 206 mm [[9]] camera is mounted 132,4 ± 0,5 cm higher and 89 ± 0,5cm behind the reference point.

6. CONCLUSION

The cases are also development stages of the software and when the next processing method has better results then previous I use only the next method.

The analysis described in case1 is not used any more; analysis in case3 can also process images with ambient scene illumination with better results than case1. The only reason to describe this analysis is that it contains some ideas which can be utilized in other areas of computer vision.



steering:6 confidence:85

Figure 11 Program output

References

[1] http://bmrc.berkeley.edu/frame/research/mpeg/mpeg_overview.html (Description of MPEG standard)

[2] http://www.cplusplus.com/ (C++ code examples, commands overviews)

[3] http://en.wikipedia.org/wiki/Covariance (Covariance mathematical formulas and features)

[4] http://www.darpa.mil/grandchallenge/index.asp (Darpa - Grand challenge competition homepage)

[5] http://www.cs.cf.ac.uk/Dave/Multimedia/node252.html (Motion vector description)

[6] http://www.mathworks.com/ (Matlab code exchange website)

[7] http://research.graphicon.ru/machine-learning/gml-ransac-matlab-toolbox-2.html (Ransac toolbox and example)

[8] http://forums.nvidia.com/index.php?showtopic=28511 (information about NVidia GeForce graphics cards)

[9] http://www.automobilemag.com/am/1998/land_rover/discovery/specifications.html (vehicle specification)

[10] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FISHER/RANSAC/ (RANSAC specification)

[11] Distinctive Image Features from Scale-Invariant Keypoints DAVID G. LOWE Computer Science Department, University of British Columbia, Vancouver, B.C., Canada Lowe@cs.ubc.ca Received January 10, 2003; Revised January 7, 2004; Accepted January 22, 2004

[12] IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 5, NO. 4, DECEMBER 2004 309 Color-Based Road Detection in Urban Traffic Scenes Yinghua He, Hong Wang, and Bo Zhang