Implementation of autonomous line following robot using FPGA and camera module

Introduction

Something for introduction

Related Works

Exist to basic approach to control vehicle movement along a line by camera. The main difference between them is camera location. In the first approach camera placed in the bottom of the vehicle [link]. The main benefits are received image is vertical and fact that line will not be closed by another object. In the second approach camera captures line segment in front of the vehicle[link]. Benefit of this realization is ability to analyze furthest line segment for changing vehicle behavior. For example, reducing speed before turn.

Algorithm for image processing

1. **Motivation for a new algorithm**

   On-board FPGA memory capacity is limited. On some models, for example Intel MAX 10 FPGA 10M50, the amount of memory can reach

hundreds of kilobytes. Nevertheless, this is not enough for image processing. A common solution is integrating external DRAM, but this approach increase power consumption and size of circuit.

On the other hand, exist another approach to overcome memory limitation. The key idea of this solution based on the mechanism of image transmission from the camera. FPGA receive frame pixel-by-pixel way. Therefore we can apply an algorithm that will handle vehicle control on pixel level. In this approach, no memory is required to store the image. Also using pixel position and different regulators we can reach complex algorithm's behavior to satisfy problem requirement.

We called developed algorithm the "pixel-flow algorithm", due to it handle vehicle control on pixel level.

2. Main algorithm idea

In order to start deriving an algorithm, we determine type of mathematical model that can be used as algorithm basis. The core computational function represents as $f : \mathbb{N}^3 \longrightarrow \mathbb{N}$, where the argument consist of two dimensional pixel position and pixel value. Due to pixel-by-pixel handling, we use iterative approach for control signal counting:

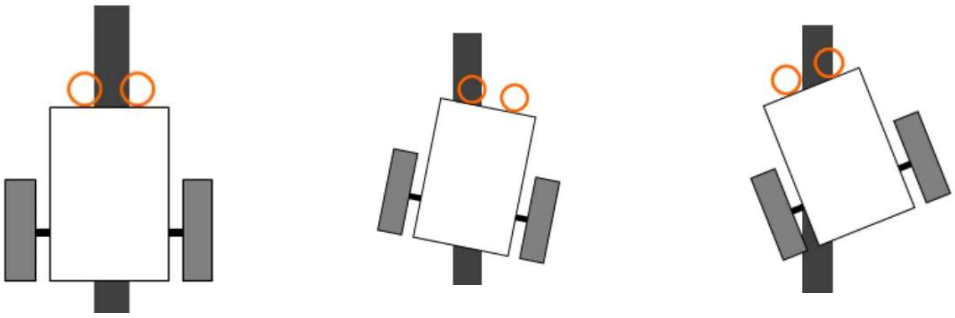
$$C_n = C_{n-1} + f(\mathrm{x},\ \mathrm{y},\ \mathrm{c})$$

where C control signal value and (x, y, c) represents x, y axes position and pixel value.

General algorithm idea is developed control system with two light-sensitive sensors. Example of vehicle with two light sensors is shown on figure 1. Picture 1.a represents position when no aligning effort is required and pictures 1.b and 1.c represent positions when aligning effort will depend on the difference of sensors value.

We generalize two-light sensors approach on array of pixels, where the left and right sides of image represents left and right sensors respectively. The difference between these sides used as main control signal.

Whether a pixel belongs to a line is determined based on the fact that the surface surrounding the line significantly differs in color. Therefore, we use threshold binarization in order to determine belonging the current pixel to the line. Also binarization generate noise that approximately the same in the entire frame. However noise reduced, because resulting control signal depends on the difference of frame sides.



a) b) c)

Figure 1: Sensors

To count each side weight we use distance of black pixels from central vertical line and multiply it by two coefficients:

1. Camera perspective distortion correction (Affine transformation)
2. Coefficient that reflect the distance from the vehicle. (farther away part of the line has smaller weight)

The ideological structure of the project represents on figure 2. The camera forms a frame and sends it to the FPGA pixel-by-pixel. During image transfer pixel-flow algorithm compute control signal. When the last pixel received, proportional regulator changes the control signal and send it to the motor driver. To control a motor power, we use pulse width modulation (PWM) technique.

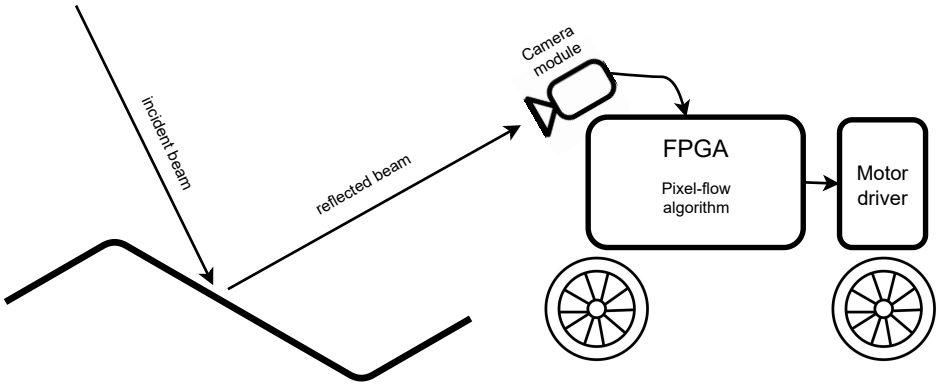


Figure 2: image

The resulting vehicle control algorithm consist of following steps:

i. (Initialization) Set zero to the control signal registers
ii. Wait the frame receiving starts
iii. Receive the pixel and update pixel position
iv. Threshold pixel value
v. Compute camera perspective distortion and distance dependent coefficients
vi. Update control signal value
if current pixel the last go to step vii, otherwise iii
vii. Update control signal using proportional regulator
viii. Set control signal on motor driver
ix. Set zero to the control signal registers and go to ii step

When we launch FPGA, it initialize camera communication protocol and resets coefficients registers to zero. As soon as camera is ready to send new frame, it rise high level signal on separate communication line. During frame pixel-by-pixel transferring FPGA determine belong current pixel either to left or right frame side and then compute coefficients and update control signal register. When frame transferring is

finished, FPGA compute difference of frame-sides coefficients. Proportional regulator converts received difference into 8 bit value and sends it to the PWM module.

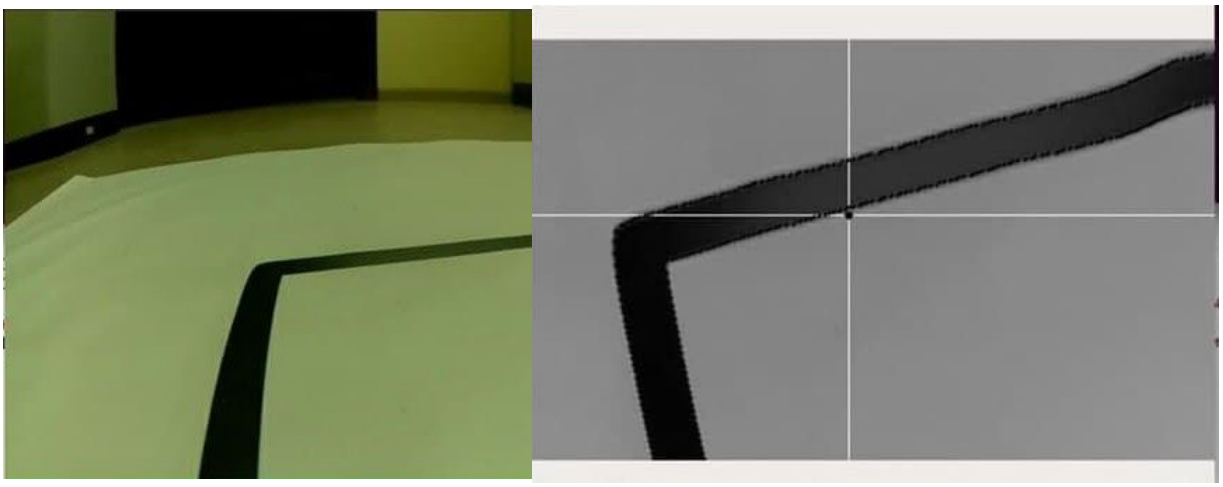
3. Algorithm math details

3.1. Basis changing

To simplify computations coordinate system of pixels counter should be changed to another one based in middle of the lower boundary of the image. For pixel counting we use a coordinate system based in the top left corner of the image and y-axes pointing down, x-axes directed to the right. Equation (1) reflects the transition to a new coordinate system

$$\begin{cases} x_{\text{new}} = |\text{width\_of\_image}/4 - x_0| \\ y_{\text{new}} = -y_0 \end{cases} \tag{1}$$

where x_{new} and y_{new} are coordinates in a coordinate system based in the center of the image bottom.



a) b)

Figure 3: Affine

3.2. Affine transformation

The camera inclination leads to occurrence of perspective distortion. To correct perspective distortion we use affine transformation. The influence of affine transformation shown on figure 3, where figure 3.b the result of correction of
figure 3.a.

In our case transformation shapes: isosceles trapezoid to rectangle, where the bases of the trapezoid are parallel to the bottom of the frame. Therefore we have the simplest case of transformation, where the bottom base of the trapezoid is almost two times bigger than the upper base. In more complex cases, transformation shape can represent an arbitrary convex quadrilateral. We can compute distortion coefficient of the each pixel using the following formula:

$$\left(1 + \frac{y}{430}\right) \tag{2}$$

where 430 is height of our image.

3.3. Deriving deviation coefficient

Also we introduce a coefficient that reflect the distance of the line segment from the vehicle. Empirically it was found that the exponential function with a range of coefficients from 1 to 0.3 is most effective, where 1 and 0.3 refer to bottom and top respectively. For computing exponential function on FPGA we use Maclaurin series:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \ldots = \sum_{i=1}^{\infty} \frac{x^n}{n!},\ |x| < \infty \qquad (3)$$

with sufficient accuracy, we can only use the first four terms.

Deviation coefficient formula:

$$m(x) = 1 - \left(\frac{x}{405}\right) + \frac{\left(\frac{x}{405}\right)^2}{2} - \frac{\left(\frac{x}{405}\right)^3}{6} \qquad (4)$$

Coefficient "405" is used for values normalization
e(0.3, 1) to x(-1.066, 0), 430 = 405*1.066

3.4. Resulting coefficients formula

Equation (5) represents the resulting formula for affine transformation and deviation correction coefficients.

$$\left(1 + \frac{y}{430}\right)\left(1 - \left(\frac{x}{405}\right) + \frac{\left(\frac{x}{405}\right)^2}{2} - \frac{\left(\frac{x}{405}\right)^3}{6}\right) \qquad (5)$$

3.5. The choice for algorithm parameters

Since Verilog HDL is hardware description language it does not have effective division operator. To overcome this challenge we replace division by shifting operator. Example of replacing is shown below:

$$\frac{1}{43} \approx 0{,}02326$$

$[A]$ - rounding to nearest integer

$$\frac{1}{43} = \frac{2^n}{43 \cdot 2^n} \approx \frac{\left[\frac{2^n}{43}\right]}{2^n}$$

example: $n = 7$

$$\frac{2^7}{43} \approx 2{,}9767 \approx 3 \Rightarrow \frac{1}{43} \approx \frac{3}{2^7}$$

$$\frac{3}{2^7} \approx 0{,}02362$$

The second challenge is normalizing the difference of frame's sides before sending it to the proportional regulator. To evaluate the required size of the register we compute the maximum side coefficient value:

$$\int_0^{430}\int_0^{320}\left(1+\frac{y}{430}\right)\left(1-\left(\frac{x}{405}\right)+\frac{\left(\frac{x}{405}\right)^2}{2}-\frac{\left(\frac{x}{405}\right)^3}{6}\right)xdxdy \approx 1{,}96968 \times 10^7 \tag{6}$$

Required register size:

$$\log_2 19696800 \approx 24{,}23 \Rightarrow 25 \tag{7}$$

For normalization of coefficients values we use the following algorithm:

i. Shift to the right by 16 bits: (0;19696800) ⇒ (0;300)
ii. limit the value to 255 if it exceeds

3.6. Motor controller

Motor controller module represented by proportional regulator and PWM modules. The motor controller perform two main operation:

i. Received control signal is changed by proportional regulator.
ii. PWM module generate control signal that is fed to motor driver.

In order to get a stable work of the algorithm should be noticed some environment parameters such as width of the line, lighting, power of the motors and vehicle inertia. Therefore proportional regulator is used for tuning the sharpness of algorithm and to handle environment parameters issues.

Implementation

In our project we employ following devices and software tools:

1. Cyclone IV FPGA board based on Altera EP4CE6E22C8N chip of low-cost and low-power Cyclone IV FPGA device family. The board appearance shown on figure 4

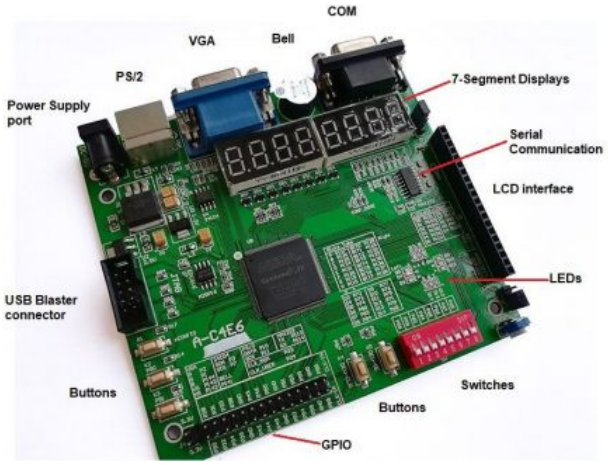


Figure 4: image

2. OV7670 Camera module based on single-chip VGA camera and image processor in a small footprint package. It has resolution 640x480 and

Figure 5: image

8-bit serial bus for image transferring, providing up to 30 frames per second. Camera appearance shown on figure 5.

3. Motor driver L298N. It has 2 pins to direction control and one speed control pin each of two motor output. Appearance of driver shown on figure 6.

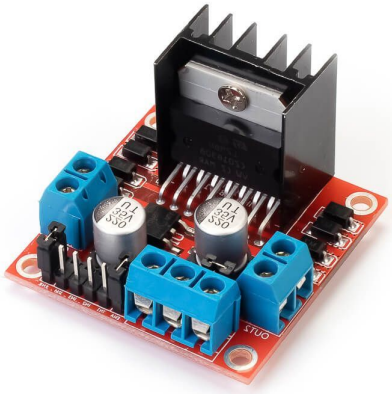


Figure 6: image

4. Software development environment Intel Quartus Prime 20.1V (Lite Edition). Program developed on Verilog HDL programming language. Verilog HDL is one of the Hardware Description Language that uses modules as basic blocks for schematic description.

The appearance of the vehicle is shown in the figure 7.

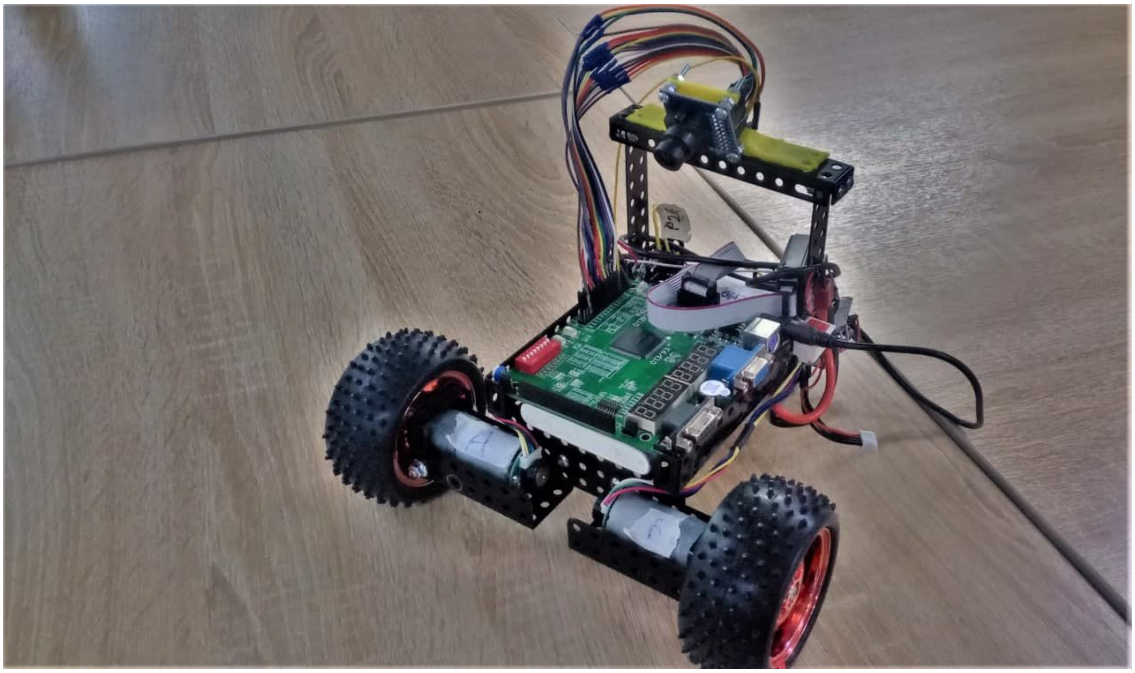


Figure 7: image

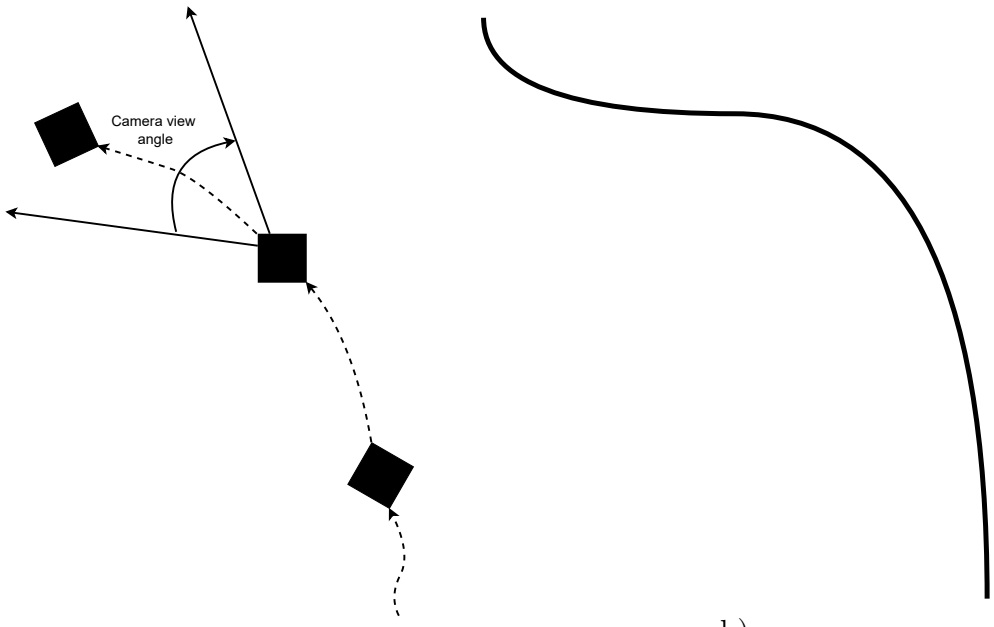


Figure 8: types of guide marks

Evaluation

Derived algorithm was tested using two types of tags shown on figure 8. First type are represented on figure 8.a. The only restriction on this type is the next tag should lie in the camera view angle. The size and distance between tags depend on vehicle configuration. The figure 8.b represent example of continuous tags that also can be used for vehicle direction. The maximum curvature of the trajectory is determined from the required speed of movement and the inertia of the vehicle.

Also, to assess the resource consumption of the algorithm, we compiled the pixel processing module separately. Required number of logical units for algorithm processing is 338. Cyclone IV FPGA has 6,272 logical units, hence the amount of resources spent is about 5 percents.

Future improvements

One of the main algorithm parameter is the binarization threshold. It depends on the color of line, lightning and sensitivity of the camera. We going to implement additional algorithm in order to automatize threshold determining. An algorithm may use first few frames for computing stable threshold

value.

Also, due to the pixel processing is done independently, it possible to handle several pixels simultaneously. The throughput of this approach will be determined from the number of parallel processing branches. To implement this idea, we also need to create a queue for incoming pixels.

Conclusion

Developed algorithm based on the assumption that