# **3D stereo camera – ground robot tracking algorithm**

Project done by : Ran \_\_\_\_\_ , I.D : \_\_\_\_\_\_\_\_\_\_\_\_

Semester \_\_\_

Table of Contents:

[**3D stereo camera – ground robot tracking algorithm** 1](#_Toc457050975)

[Requirements 1](#_Toc457050976)

[Background for the project 1](#_Toc457050977)

[The details about the parts of the problem solution 2](#_Toc457050978)

[Implementation 3](#_Toc457050979)

## Requirements

Goal is to track a moving target using Stereo vision on a mobile robot. main scenario phases:  
1.  Scenario starts with static background.  
2.  Target is acquired using motion detection (background subtraction, or other)  
3.  Robot moves towards target as long as the target is more than some pre-defined distance away  
4.  Robot steers towards target as needed, in order to not lose it.  
  
Research (Detailed report on all these):  
1. Study how low can the image resolution be before tracking is lost  
2. Measure the processing speed (frames per second) at that resolution  
3. Study how fast can a target move without being lost  
  
Submission contains:  
1. Report  
2. Presentation  
3. Video of demo  
4. Live demo

## Background for the project

The project includes few issues to consider:

3D stereo camera – it is consisting on two regular usb cameras, installed in parallel with each other, in a constant distance. This structure can be calibrated in advance. This construction can give us more information from the images – it gives us the distances of the observed objects.

How to calibrate each camera, and the construction of 2 cameras together ? it can be done offline , before the mission itself. I will explain later about the calibration details.

the moving object to track (called the Target) – how to measure a moving object while the robot is moving ? that’s why the target identification is done while the robot is standing. I am using the ‘Background Sunstration’ function to find the target.

Tracking : the tracking after the target starts after the identification phase. The tracking is done using only 1 camera (left as arbitrary decision) , while from time to time, a distance data from the stereo cameras structured, is fused into.

## The details about the parts of the problem solution

3D stereo :

* Calibration
* Disparities + Depth map

Calibration: there is a given procedure for conducting the calibration of both of the cameras.

The calibrations are done for each of the cameras separately (intrinsic), and then together (extrinsic), to find the physical distance between them.

The outcome of the calibration procedure is extrinsic and intrinsic data of each camera. As it is known as a standard method of camera calibration.

Using the 2 images we get at each time step, we can calculate disparities – these are the differences between feature points in one frame (usually the left image) to the other. From that, and given the already prepared calibration matrices of both of the cameras, we can ‘rectify’ the images.

That is – aligning the images relative to each other, such that any pixel in any row in 1 image, will be in the same row in the other image – meaning the differences between the images will be in the pixels in their horizontal location (coordinate), and that’s it.

From these disparity data , we can get the depth map of the scene. And also construct 3D points cloud. All relative to the point of the camera base location.

To get the real global location of the objects we need to consider the relative changes from time point to the next, and do some integration and estimation to get the global positioning and velocities of the robot’s platform and the tracked object. – this part is not included in this project assignment.

A typical action scenario of the system:

1. Robot in static position. Waiting. Watching.
2. Identify movement. The object (person) should stand in front of the robot camera and wait ‘a moment’. The robot will identify the target and lock on it.
3. Tracking will begin – the person will be followed to where he is going.

The other part of the project (not described in this report) is the ability of identify and learn the target, during the tracking, in order to allow re-identification when it comes back from occlusion.

Target identification

In the first stage, the robot is static. It observes the scene for changes. Any object moving in the Field-Of-View will be considered as target.

The algorithm method for use, in this phase, is : Background Subtraction.

Criterions for locking on object :

The object siluhete area is bigger than some threshold parameter.

Parameters for Bckng.Subs initialization :

Remembering 150 frames – in order to get a firm background model to relate to.

More frames are better, but we must consider the run-time element. The FrameRate is 15-30, depend on other properties we want from the system.

Movement identification and locking should be les then 5 seconds.

Tracking

After the movement identification and locking has been done – we must identify that object. meaning – calculating and defining some special property that will extinct this object from the background and other objects in the scene. And by this property we conduct the tracking – a search algorithm is used to look for the same identity feature, inside the next camera frame. This part is done mainly by only one camera, no need for two. But on lower frequency – we do use the 3D stereo setting ,to verify distances to the object.

There are many known algorithms for object tracking in videos- I’ve considered some of them:

Mean shift, Cam shaft, TLD, …

My choise for the Tracker is a simple one – by using optical flow.

1st step is identifying ‘good features to track’, by using

## Implementation

**Compilation:**

The source code enables to run the code on a stand-alone computer, with the stereo camera connected( by usb) , or running it from the robot itself. The relevant setting is done by a **compilation flag named ‘COMPILING\_ON\_ROBOT’.**

Which allows the compilation and cooperation of the file ‘pwm.h’ ,and operation of the wheels commands.

**Movement identification:**

Using the object of class ‘BackSubs’ that is defined in ‘BackgroundSub.hpp’ . it incorporates the object of ‘BackgroundSubtractorMOG2’ defined in the core files of OpenCV , in ‘C:\OpenCV\build\include\opencv2\videobackground\_segm.hpp’.

There are several object types of **BackgroundSubtractor** in OpenCV. I chose to use the MOG2 type, because it should be a bit more accurate (\*1). It is initialized with certain parameters. I chose the followings:

History = 300 (which is up to about 10 seconds of keeping history, for 30 FPS video source)

Threshold = 32 (minimum squared Mahalanobis distance between the pixel and the background model, in order to show as foreground)

detectShadows = false (not considering those, only to get a better run-time performance . result is accurate enough for this application)

This object is initialized with the program startup.

The main algorithm is initialized with ‘system\_status’ on ‘INITIALIZING’;

Sometimes when starting the program, the MOG object returns result like that all the frame is at move. Only when the MOG result is a blank image – the system goes to the next state of ‘STANDBY’.

From here the next identification of some movement is treated as moving target, according to the following few filters and conditions:

On the raw result of the MOG object, I implement :

threshold (foreground, foreground, 128, 255,THRESH\_BINARY);//28,128,198

medianBlur (foreground, foreground, 3);//9

erode (foreground, foreground, Mat());

dilate (foreground, foreground, Mat());

in order to smooth the image and get accurate segmentation.

It should return result of magnitude of moving target, and it’s points to identify and track .

**Target Identifiation:**

Traacker object

Processes summary :

|  |  |  |
| --- | --- | --- |
| #1 | Type |  |
| Image handler | thread | Captures images and store them |
| Main | main process |  |
| Disparity | Thread | Calculates disparity for given pair of images |
| BackgroundSubs | regular process | Calculates foreground, by the left input image |
| GUI handler | regular process | Display images and extra graphics |
| Feature tracker | regular process | Keeps data for the tracked object, by feature points. |
| - | - |  |
| - | - |  |

System state flow diagram

Initialization

אתחול – הרובוט במצב סטטי

Standby

מצב המתנה –

ע"י BackgroundSubstraction, מזהה תנועה ע"י השוואה למודל רקע.

מוגדר ע"י מספר פרמטרים שהתכנסתי אליהם. הם ניתנים לשינוי. (זמן המתנה, זמן למידה וכד').

Moving Target acquisition 1/2

זיהוי תנועה – רכישת מטרה ע"י למידת Feature Points של איזור המטרה.

המטרה היא התוצר של השלב הקודם.

Moving Target acquisition 2/2

חישוב STEREO. מקבלים תמונת DISPARITY ותמונת DEPTH של הסביבה.

מבצעים התאמה של ה feature points משלב קודם עם נתוני העומק. נקודות עם עומק חריג יבוטלו. תחושב מידת עומק ממוצעת של המטרה. בשלב הבא יצופה מרחק דומה של המטרה.

Target Tracking

עקיבה אחר המטרה ע"י זיהוי תנועת ה feature points הקודמים , בתמונה הנוכחית. תוך סינון ע"י נתון עומק הדומה למסגרת הקודמת.

במידה ואחוז ההתאמה גדול מ50% מבצעים השלמה ע"י איתור feature points חדשים באיזור המטרה החדש (bounding box).

במידה ואחוז ההתאמה נמוך מ 50% מתריעים על טיב עקיבה באיכות נמוכה.

References

1. OpenCV guide for stereo vision , calibration, and disparity calculations :

“[Learning OpenCV Computer Vision with the OpenCV Library](http://shop.oreilly.com/product/9780596516130.do)” , By Gary Bradski, Adrian Kaehler, 2008. Chapter 12.

1. OpenCV general tutorial about functions for 3D reconstruction:

<http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html>