3-14-13-Path-2019-3-26.ppt

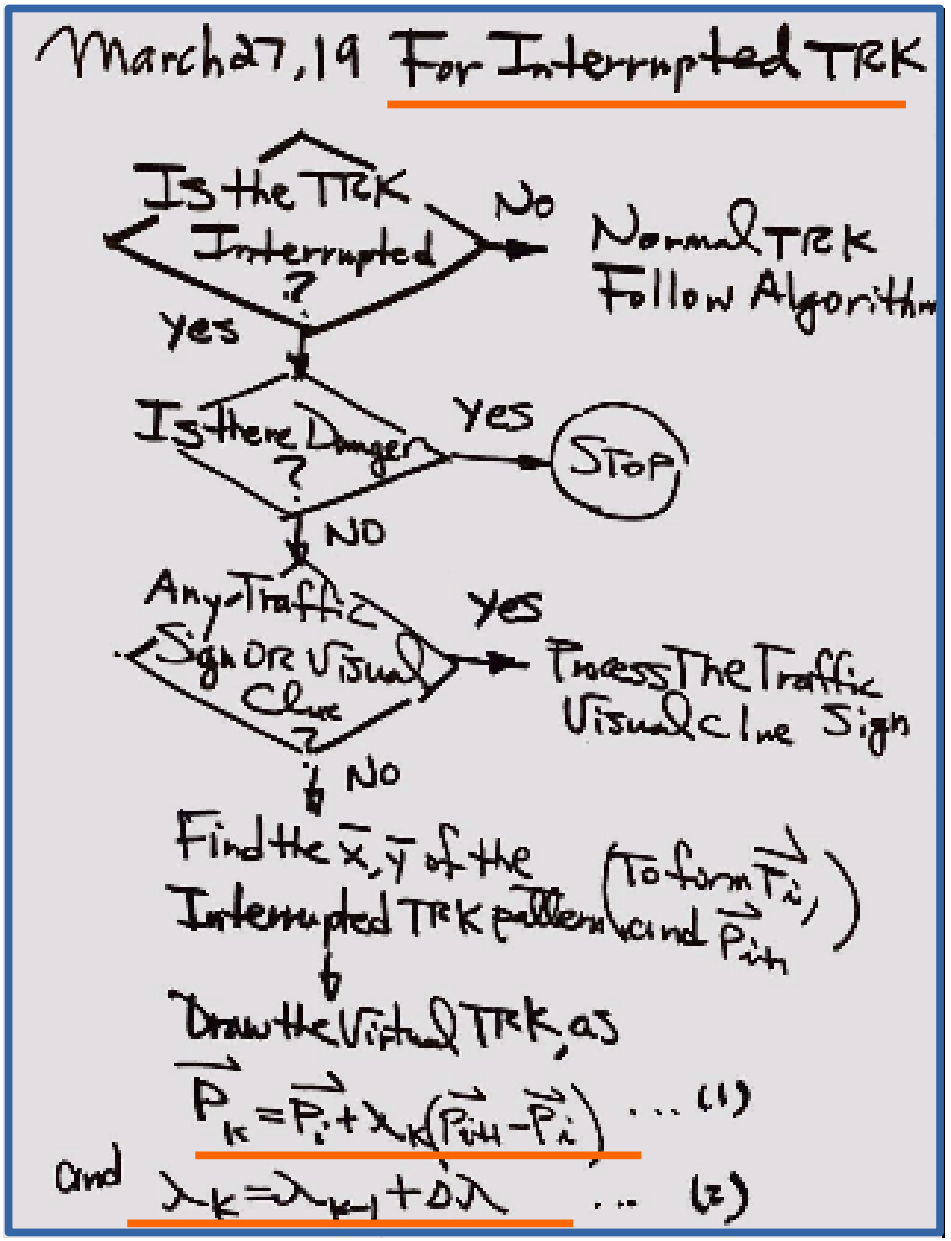
CTI One Corporation

Version: x0.1
Date: Mar. 26, 2019
Project Lead: Harry Li, Ph.D.

Group Leaders:

Team members:

Mar-27-2019 Path (1)



Mar-27-2019 Path (1)

March 27 (Wed) 19. Path Planning 1/ 7+L

CAT Ia: TRK Continuous

Ib: TRK Discontinuous No Obstacle

CAT IIa: Point-Cloud Input: Point-to-Point OR Virtual Path

Path (Fig.2) Algorithm

1° Find Feature Points $\{\vec{P}_i\}$;

2° Selection of Destination Point $\vec{P}_{Di}$, Starting Point $\vec{P}_{Si}$

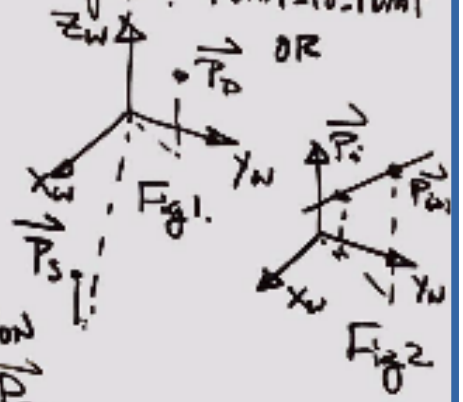
3° Form the Shortest Path: $\vec{P} = \vec{P}_{Si} + \lambda(\vec{P}_{Di} - \vec{P}_{Si}) \cdots (1)$

4° Drive on the Path, Incrementally $\lambda \in [0, 1]$

$\lambda_0 = 0$, then $\lambda_{k+1} = \lambda_k + \Delta\lambda$,

hence, $\vec{P}_{k+1} = \vec{P}_{Si} + \lambda_{k+1}(\vec{P}_{Di} - \vec{P}_{Si}) \cdots (2)$

CAT IIb: Point-Cloud Input, P-2-P w/Obstacle



March 27 (Wed) 19. Path Planning 2/ 7+L

CAT IIb: Point-to-Point w/Obstacles

$\vec{P}_{O,j}$ Obstacle Point

$\{\vec{P}_{O,j}\}$ all obstacle points.

Starting Point: $\vec{P}_{Si}$, Destination Point: $\vec{P}_{Di}$

1° Check Obstacle Point, if yes, then Predestination Point

$\vec{P}_{Oj} = \vec{P}_K \cdots (1)$, $\vec{P}_{k+1} = \vec{P}_{Si} + \lambda_{k+1}(\vec{P}_{Di} - \vec{P}_{Si}) \cdots (2)$

2° Assign New Destination Point as

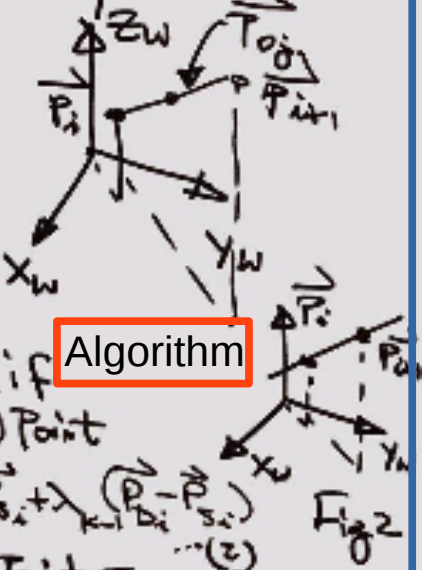
$\vec{P}_{D,i+1} = \vec{P}_{K-1} \cdots (3)$

3° Drive up to $\vec{P}_{D,i+1}$ by $\vec{P}_l$, $l = 0, 1, 2, \ldots, k-1$

4° Find New Starting Point $\vec{P}_{Si+1}$. go to Step 1°.

If End of the Drive, yes → Stop

No →



Mar-27-2019 FSM For CAT-II-b Path

10-25-2018 3-Stages Find/Reach/Pass the Door

```
bool X0_Door_Detected() { //To detect the door
        bool flag =0;
        //Algorithm to detect the door in the FoV of Zed here
        return flag;   }
bool X1_Door_Reached() { //To detect if vehicle reach the door
        bool flag =0;
        //Algorithm to detect if the door is in front of vehicle here
        return flag;   }
bool X2_Door_Passed() { //To detect if vehicle passed the door
        bool flag =0;
        //Algorithm to detect if the door is behind vehicle here
        return flag; }
//Detect obstacle, orientation adequate, clearance adequate
bool X3_Obstacle_Detected() {
        bool flag =0;
        //Algorithm to detect if obstacle is on the path here
        return flag; }
bool X4_ULTSEN_Positive_Pass () {
        bool flag =0;
        //Algorithm to detect if the L/R proximity is adequate here
        return flag; }
bool X5_Orientation_Adequate() {
        bool flag =0;
        if (ULTSEN_Positive_Pass) { // L/R proximity from  door is good
                flag = 1;
        return flag; }
bool X6_Clearance_Adequate () {
        bool flag =0;
        //Algorithm to detect if the path is clearance
        return flag; }
```

bool X0_Door_Detected()
bool X1_Door_Reached()
bool X2_Door_Passed()
bool X3_Obstacle_Detected()
bool X4_ULTSEN_Positive_Pass ()
bool X5_Orientation_Adequate()
bool X6_Clearance_Adequate ()

~/Documents/CTI0/3 项目 /3-14-AGV2000/lec/lec3-1-4-4-Path/source$PassThruDoor_1.cpp

11-6-2018 vidpat.cpp for AGV2000

Nov. 6, 2018.

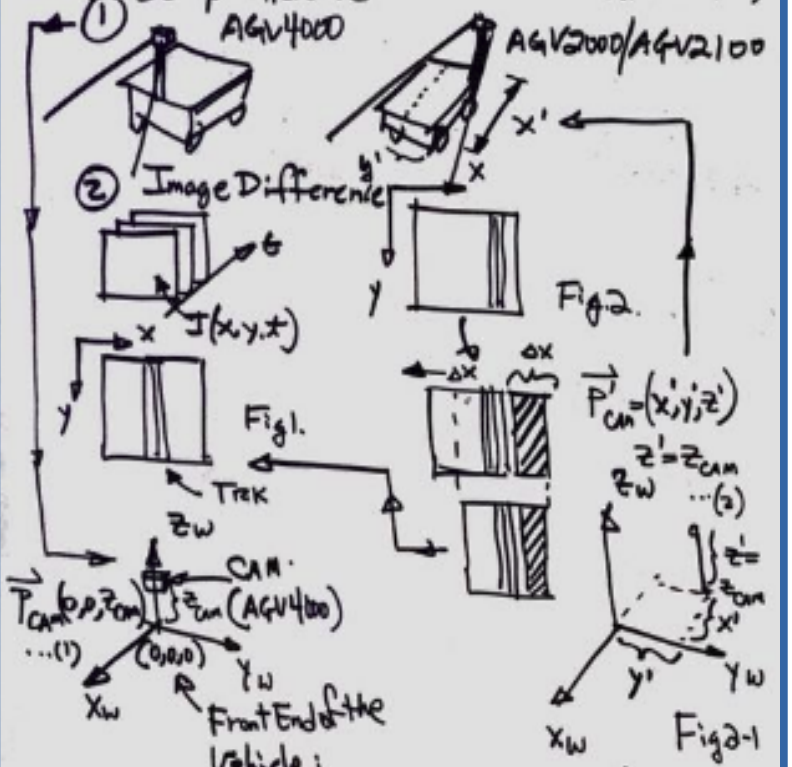
1) AGV2000/AGV2100 vidpat.cpp (Originally For AGV4000)

Setup Difference

① AGV4000

AGV2000/AGV2100

② Image Difference



$\Delta x = f_1(x',y',z')\big|_{z'=z_{cam}} \cong f_2(x',y')$ ...(3) where x', y' can be measured.

Experimentally, find Δx:

Property 1: If vehicle is centered, then its CAM must be centered; Hence, the TRK on the I(x,y) should have no offset, e.g. centered.

Verification: See Fig. 1.

Property 2. If property 1 holds good, then the center TRK will have to be located in the following Region, e.g. Centerline region. as in Fig. 3.

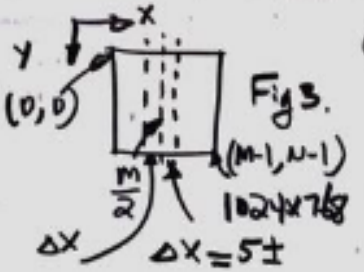


Fig 3.

#define Center_Line M/2

#define Center_Line_margin deltax

Implementation:

① Take Video from AGV2100 → Capture I(x,y)

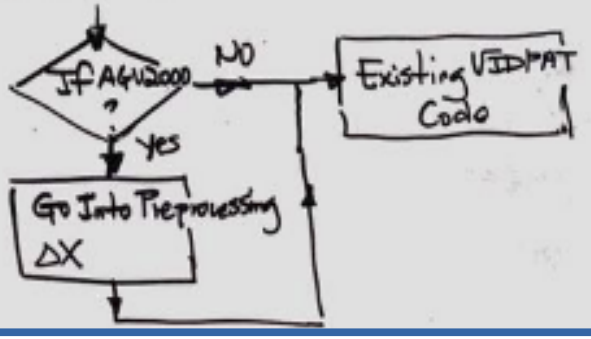
VideoCapture Cap(argv[1]); Cap → Shift Δx.

fill property 3. Satisfied;

Property 3. For off-centered CAM setup as in Fig. 2, the TRK is offset by the amount of Δx. as in Fig. 2. therefore, the following correction is made to move I(x,y) to the left by Δx, as the result, TRK will be relocated back to the Target Region.

Preprocessing Part.

#define AGV2000 1



Properties 1-3 are for TRK offset correction due to off center CAM setup