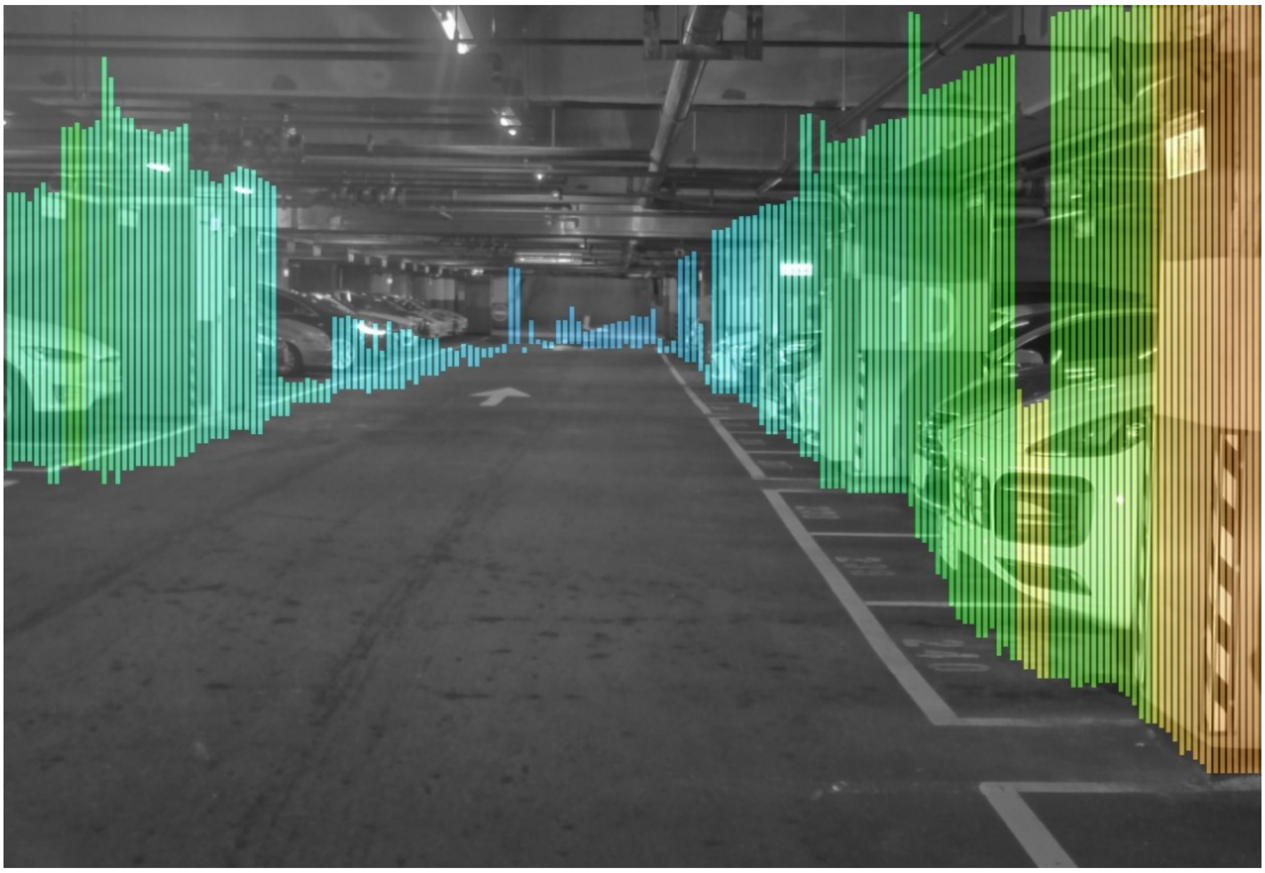
Table of Contents

Introduction

- Stixel is a portmanteau word: stick + pixel = stixel, it’s a compact form representing object.


Each stixel(stick) represent obstacle

- This sample code is dedicated to following hardware configuration, but it's not limited to it, user can find tune the software parameters based on the module under test.

Stereo camera module

	Model	Note
ImageSensor	IMX424(2pc)	4K sensor
Lens	CAR53(2pc)	

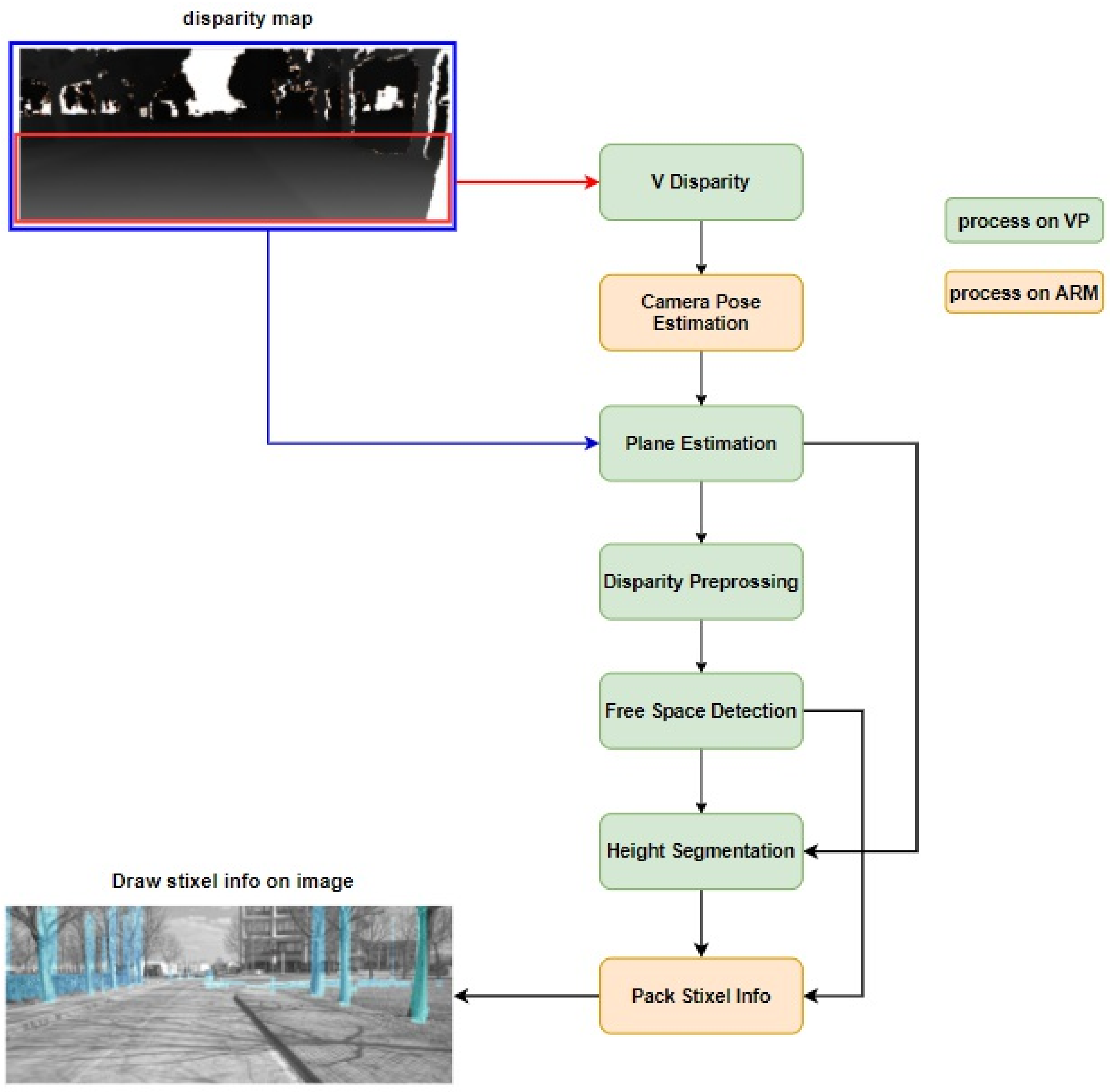
Setup

Item		Note
BaseLine	0.25(meter)	4K sensor
RectifiedFocalLength	1590(pixel)	At scale2(1920x960)

- This document walks though
  1. The theory behind the sample code.
  2. List some of its major limitation, and give corresponding suggestion.

Workflow

Following is a big picture of the sample code, detail of each block is described in later section.



Work flow of the sample code

Step Explanation

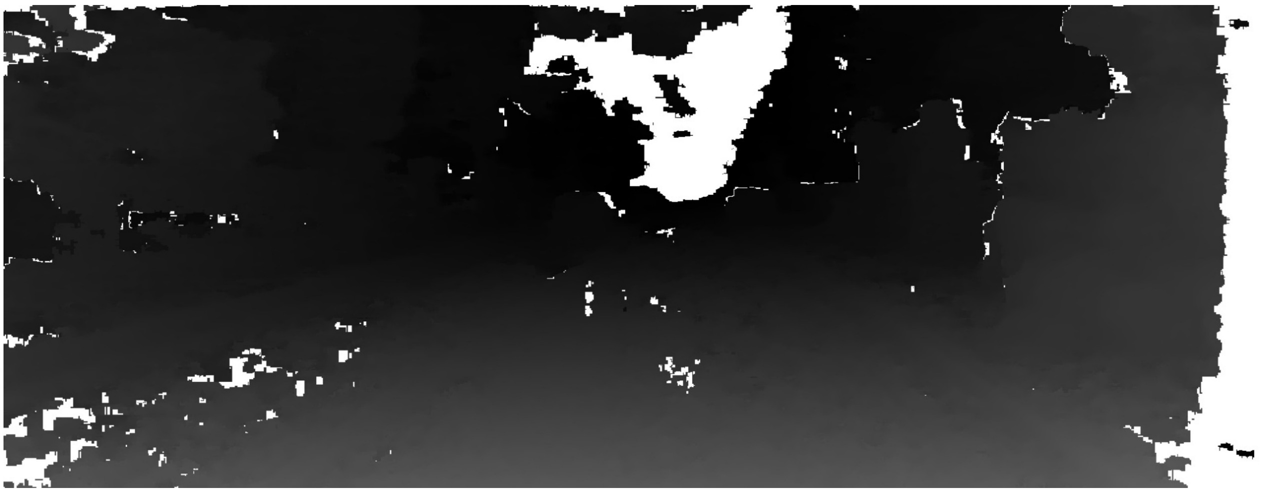
V Disparity

It take disparity map as input, and calculate v-disparity.

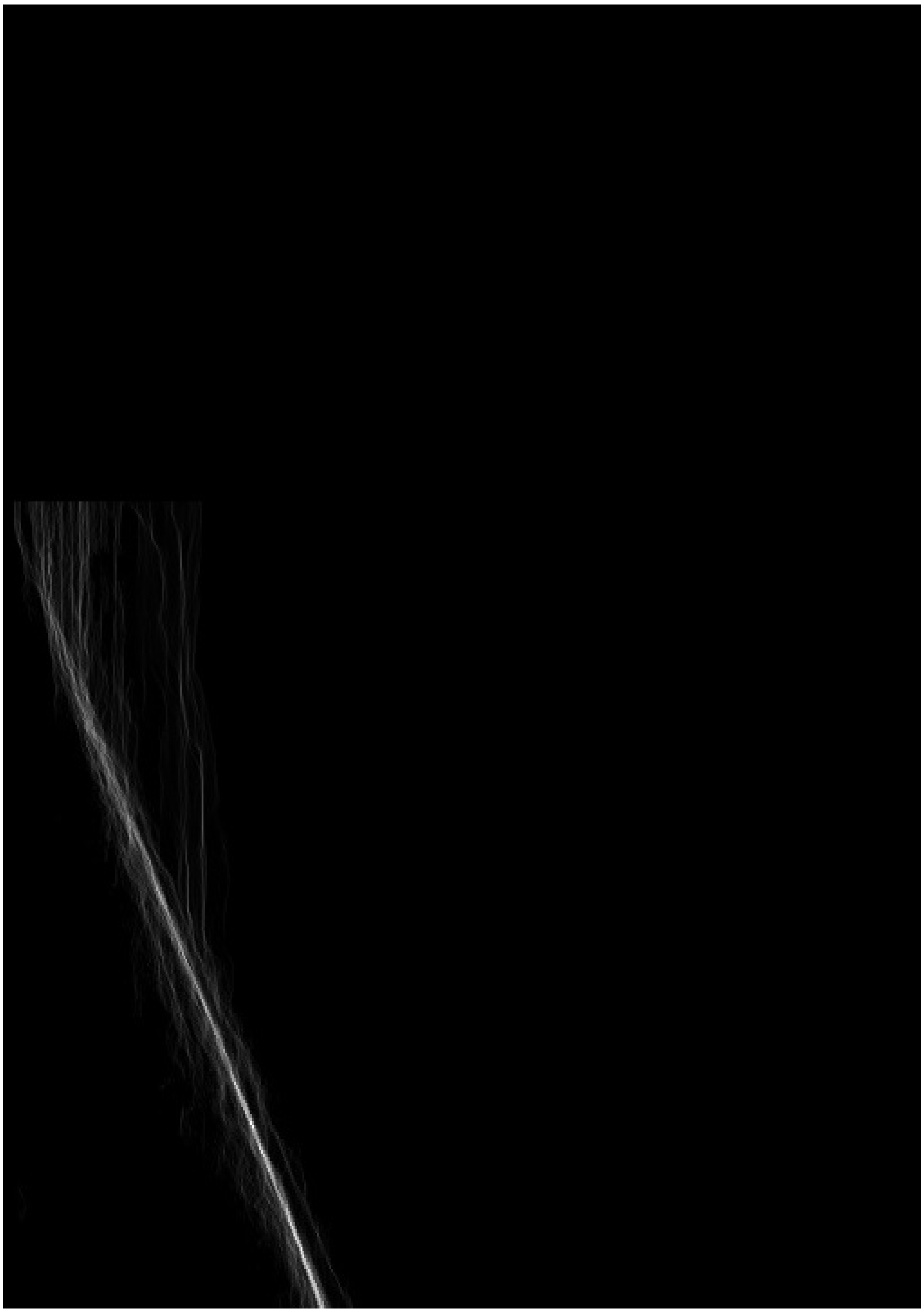
V-disparity is a common way to get sense of ground lying forward. It's a 2D map with the same height of disparity map, and width is the the max possible disparity value.

For each row in disparity map, it count the number of each appearing disparity value, and accumulate to V-disparity map.

Source image



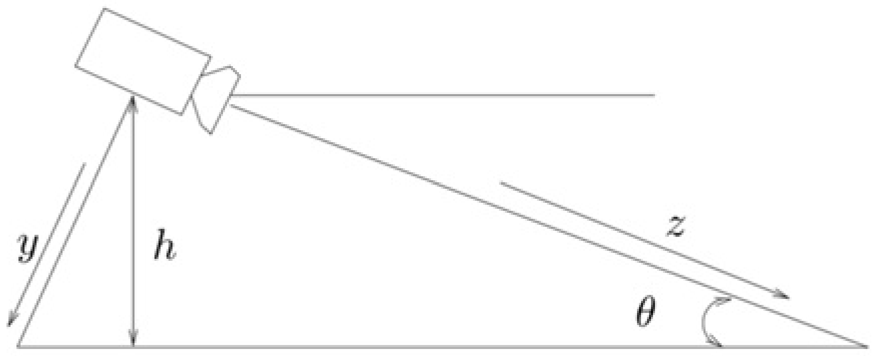
Disparity


V-disparity

With assumption that generally stereo head is put parallel to ground, one can get a relatively obvious straight line in the v-disparity map.

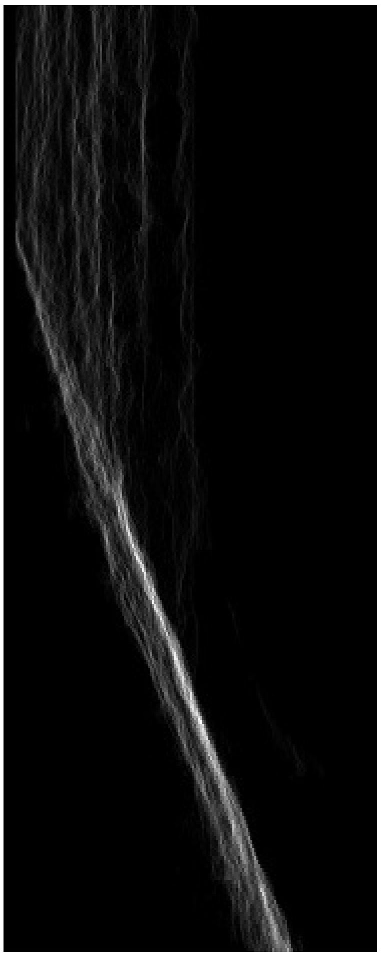
Camera Pose estimation

Once get the line equation from v-disparity map, one can derive camera pose(pitch and height), and vice versa.



Geometry of camera and ground plane

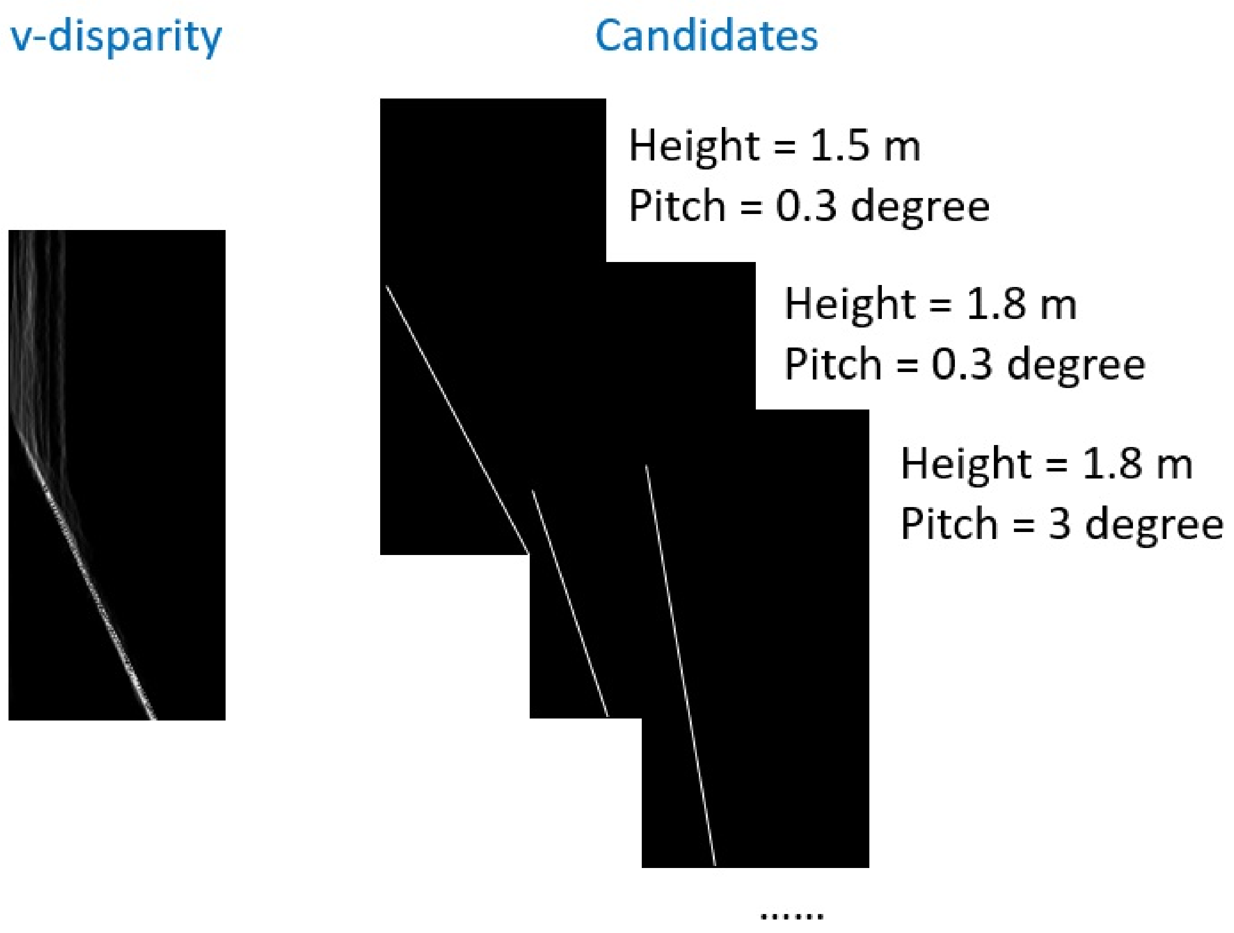
In practice, v-disparity map is noisy, so directly use line search algorithm to find out line equation may yield poor camera pose estimation.



Noisy v-disparity

So here Instead generate a bunch of candidate line equation given known intrinsic parameters and default camera pose.

In runtime it compares v-disparity map to each candidate in the pool, current camera pose should be approximated to the one whose v-disparity is the most similar.



Search for the most similar line in the pool

One prerequisite of this process is to get default pose of camera, these info can be get from

- User's private algorithm to do camera calibration
- Use auto mode to get camera pose

There are 2 modes of camera pose estimation which are suitable for different scenarios

- Auto mode

  This mode is with no prior knowledge of camera pose, it infers pose only based on v-disparity map, so it's useful to get default camera pose.

  To get best quality of the estimation, it requires parking the car on a flat ground with adequate environmental light, and large area of ground lying ahead to give sufficient clues to camera.

- Manual mode

  This mode is with defaulted camera pose, it's dedicated to get camera pose in runtime, the detail is described above.

Plane Estimation

After getting the line equation, one can generate look up table of estimated ground disparity, that is

$F(V_{Pos}) = EstimatedGroundDisparity$

It also generates following intermediate table for later computation

- H_v

  Used for calculating $ObjectScore$. For every V_{pos}, look up top position of virtual object.

- Road disparity coefficient

  Used for calculating $GroundScore$. For every V_{pos}, look up its distance to bottom of image in unit of pixel.

- ΔD_u

  Used for calculating Height segmentation.

Disparity Preprocessing

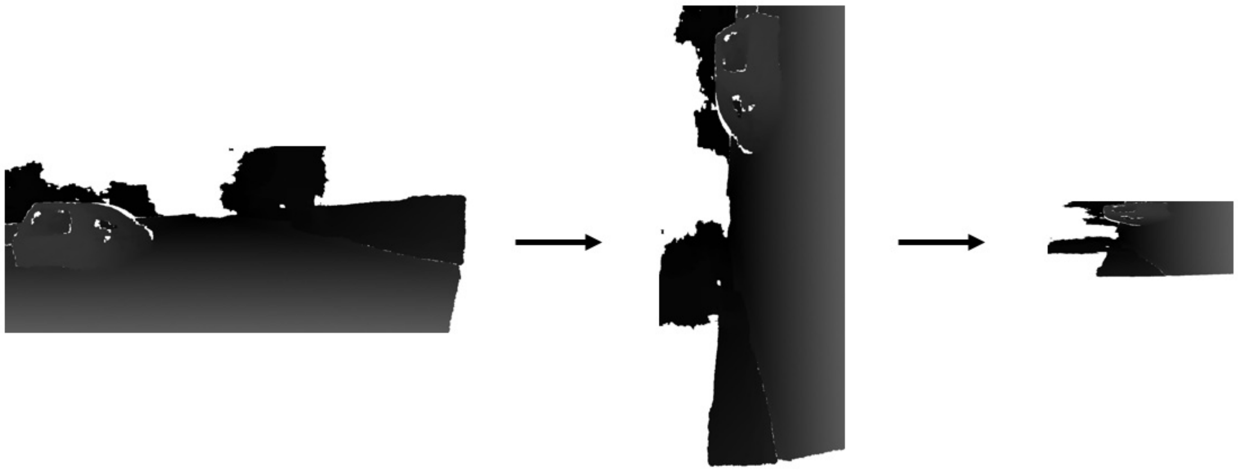
In this step, it preprocesses disparity for 2 reasons

- Transpose disparity map

  Because subsequent process just focus on each independent column of disparity map, transpose it to get cache efficiency.

- Vertical max pooling

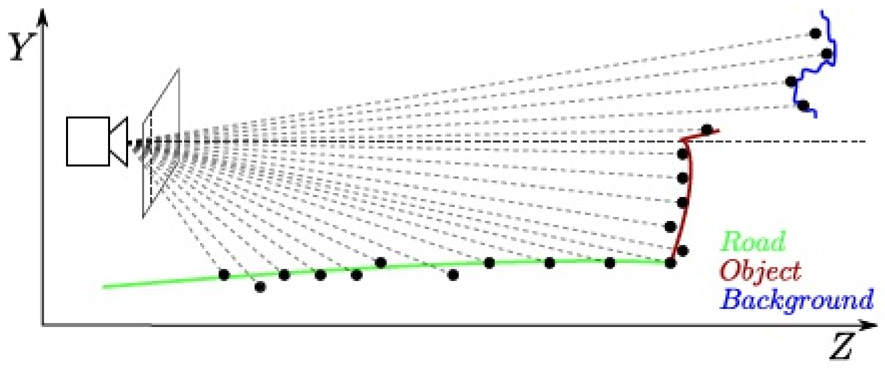
  Windows size is width of each stixel.



Left: Disparity Mid: Transposed disparity Right: Vertical Max pooled disparity

Free Space Detection

In this step, with assumption that object have constant disparity from its bottom across its face toward to camera



Typically face of object is more parallel to camera plane than ground is

For each pixel in each column of disparity map, compute its node score

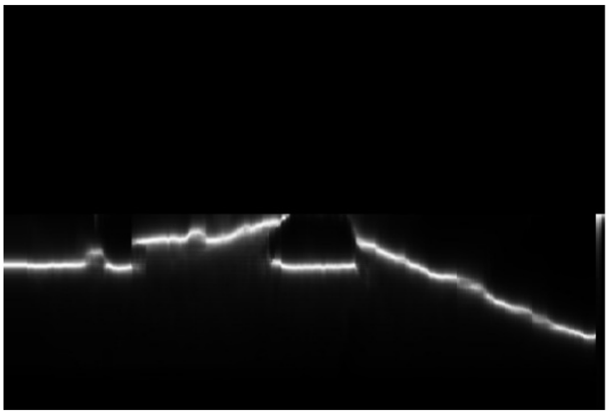
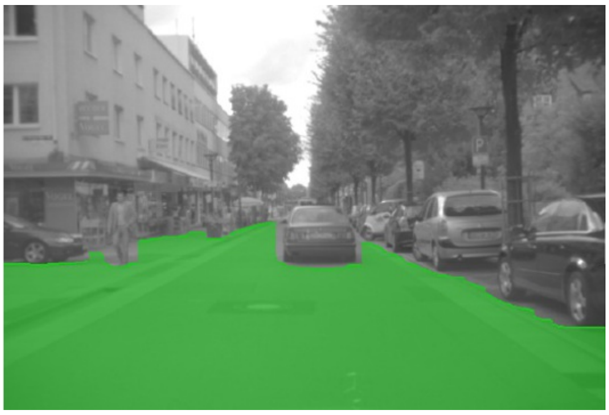
$NodeScore = \alpha GroundScore + \beta ObjectScore$

$$GroundScore(V_{Pos}) = \sum_{v=V_{Pos}}^{v=BottomOfImage} |Disparity(v) - DisparityOfEstimatedGround(v)|$$

$$ObjectScore(V_{Pos}) = \sum_{v=V_{Pos}-H_v}^{v=V_{Pos}} |Disparity(v) - DisparityOfEstimatedGround(V_{Pos})|$$

Note that H_v is object height in unit of pixel, given default camera intrinsic and extrinsic parameter, we can come up with this value on any give V_{Pos}. (see section: Plane Estimation)

Ideally the lowest node score would be on the transitioning point between object and ground

Left: Node score image. (The brighter the pixel is, the lower node score it has) Right: Result of free space detection

Height Segmentation

In this step, with assumption that there is always object appear on each column, the idea is to find best point which well divide foreground and background.

For each pixel in each column, compute membership (only consider region above free space)

$$Z = 1 - \left(\frac{Disp_v - ObjBottomDisp}{\Delta D_u}\right)^2$$

$$\Delta D_u = Disp_v - Disp_{OnFarerDistance}$$

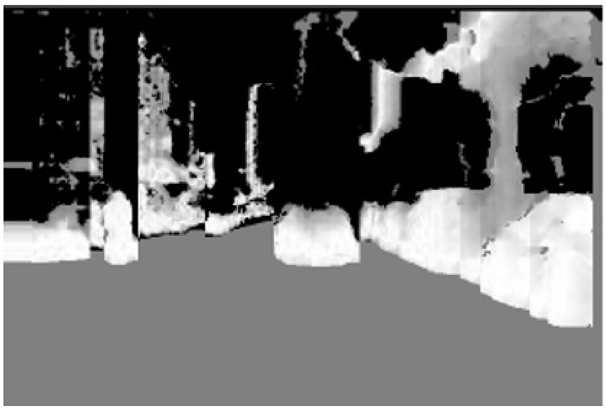
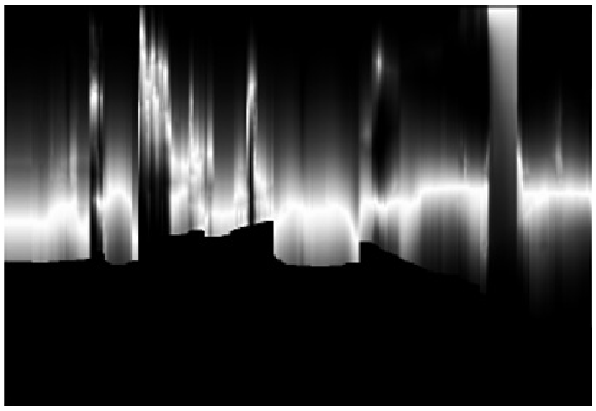
$$Membership = 2^Z - 1$$

Note that ideal foreground pixel will has $Membership = 1$, ideal background pixel will has $Membership = -1$.

For each pixel in each column, compute cost

$$C_{u,VPos} = \sum_{v=ObjBottomVPos}^{v=VPos} Membership_{u,v} - \sum_{v=VPos+1}^{v=TopOfImage} Membership_{u,v}$$

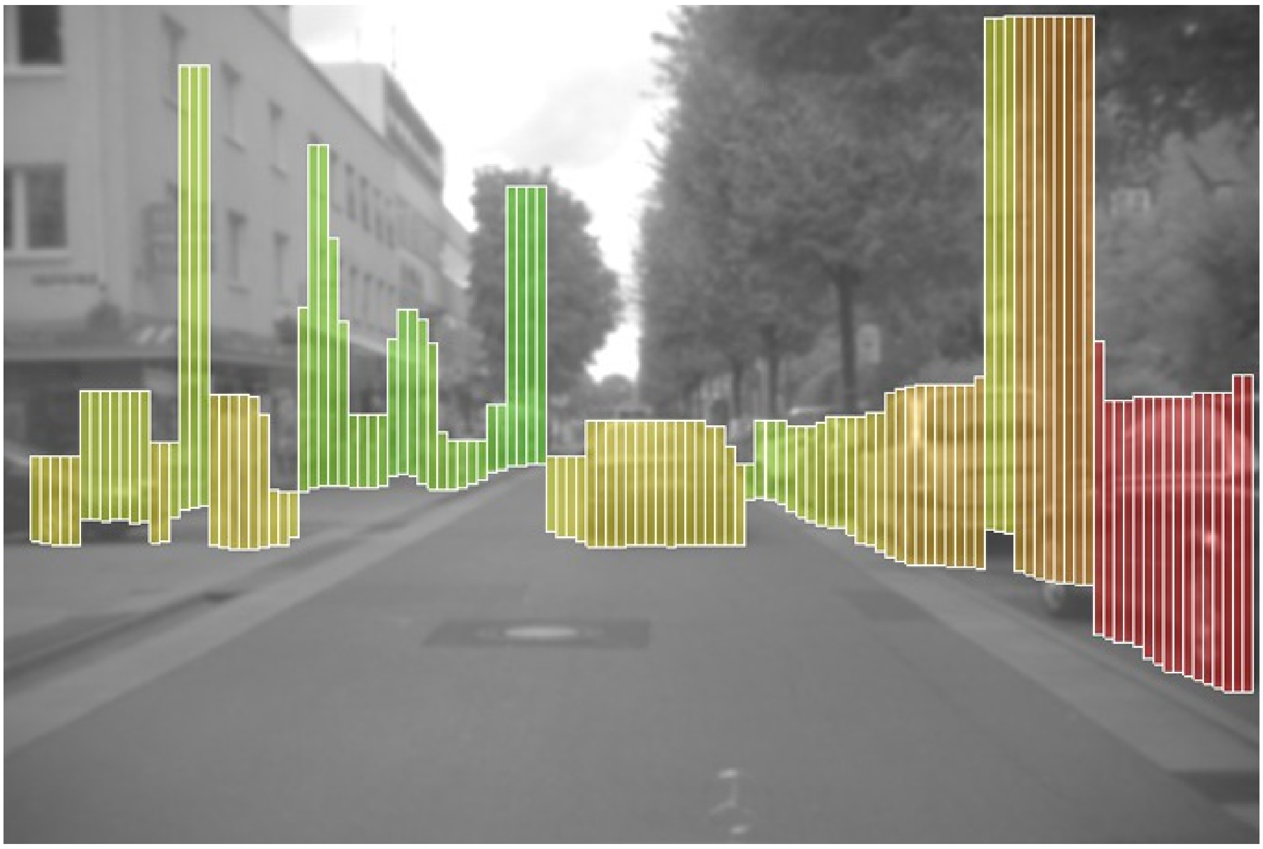
Ideally the lowest cost value($-(C_{u,VPos})$) is on the transitioning point between foreground, which is object itself, and background.

Left: Membership image, Bright pixel means its disparity value is close to the bottom of the object Right: Cost image. (The brighter the pixel is, the lower cost it has)

Pack Into Stixel

For each column we have corresponding free space and object top position, just pack these 2 info into stixel.



Visualized stixel

Parameters

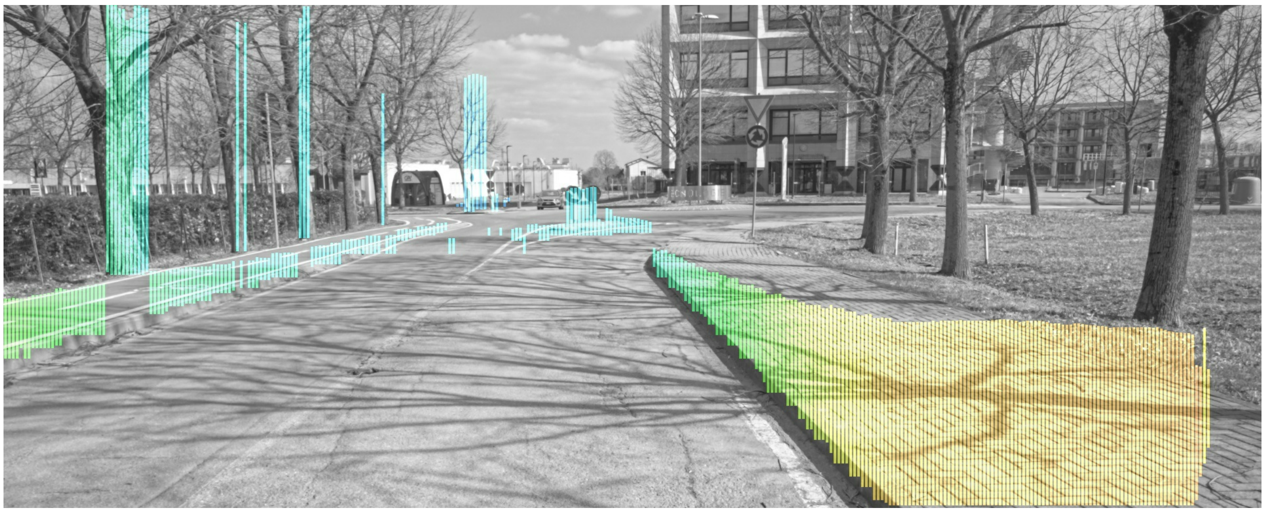
```
typedef struct {
    // Run auto mode(STIXEL_ROAD_ESTI_MODE_AUTO ) or manual mode(STIXEL_ROAD_ESTI_MODE_MANUAL).
    UINT16 RoadEstiMode;
    // Height of stixel will be forced to be 1 if disparity at its bottom is smaller than this value. (unit: pixel)
    UINT16 MinDisparity;
    // Object with a height larger than this value is more detectable. (unit: m)
    DOUBLE DetObjectHeight;
    // Intrinsic camera parameters.
    AMBA_CV_STIXEL_CAM_INT_PARAM_s IntParam;
    // Configuration of manual mode.
    AMBA_CV_STIXEL_MANUAL_DET_CFG_s ManualDetCfg;
} AMBA_CV_STIXEL_CFG_s;

typedef struct {
    // Granularity of camera pitch in candidate pool. (unit: degree)
    DOUBLE PitchStride;
    // Granularity of camera height in candidate pool. (unit: m)
    DOUBLE HeightStride;
    // Number of various camera pitch.
    UINT32 PitchSetNum;
    // Number of various camera height.
    UINT32 HeightSetNum;
    // Extrinsic camera parameters.
    AMBA_CV_STIXEL_CAM_EXT_PARAM_s ExtParam;
    // Parameters to assess quality of v disaprity .
    AMBA_CV_STIXEL_CAM_VDISP_PARAM_s VDispParam;
} AMBA_CV_STIXEL_MANUAL_DET_CFG_s;

typedef struct {
    // Horizontal ratio threshold. (range: [0.0 ~ 1.0])
    DOUBLE XRatioThr;
    // Vertical ratio threshold. (range: [0.0 ~ 1.0])
    DOUBLE YRatioThr;
} AMBA_CV_STIXEL_CAM_VDISP_PARAM_s;

typedef struct {
    // Default camera height. (unit: mm)
    DOUBLE Height;
    // Default camera pitch angle. (unit: radian)
    DOUBLE Pitch;
} AMBA_CV_STIXEL_CAM_EXT_PARAM_s;

typedef struct {
    // Horizontal focal length. (unit: pixel)
    DOUBLE Fu;
    // Vertical focal length. (unit: pixel)
    DOUBLE Fv;
    // Optical center u. (unit: pixel)
    DOUBLE U0;
    // Optical center v. (unit: pixel)
    DOUBLE V0;
    // Stereo baseline. (unit: mm)
    DOUBLE Baseline;
} AMBA_CV_STIXEL_CAM_INT_PARAM_s;
```

DetObjectHeight

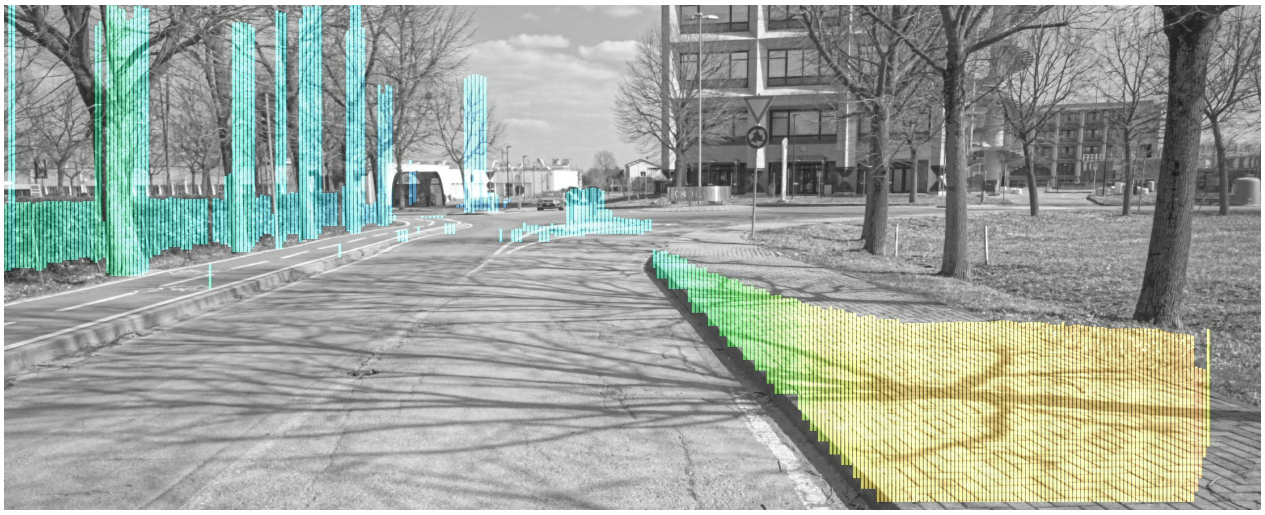
Setting this value smaller to make it able to detect shorter obstacle like sidewalk, but it would make false alarm easier especially when

- Image is noisy
- Road is not completely flat
- Stereo head is not well parallel to ground

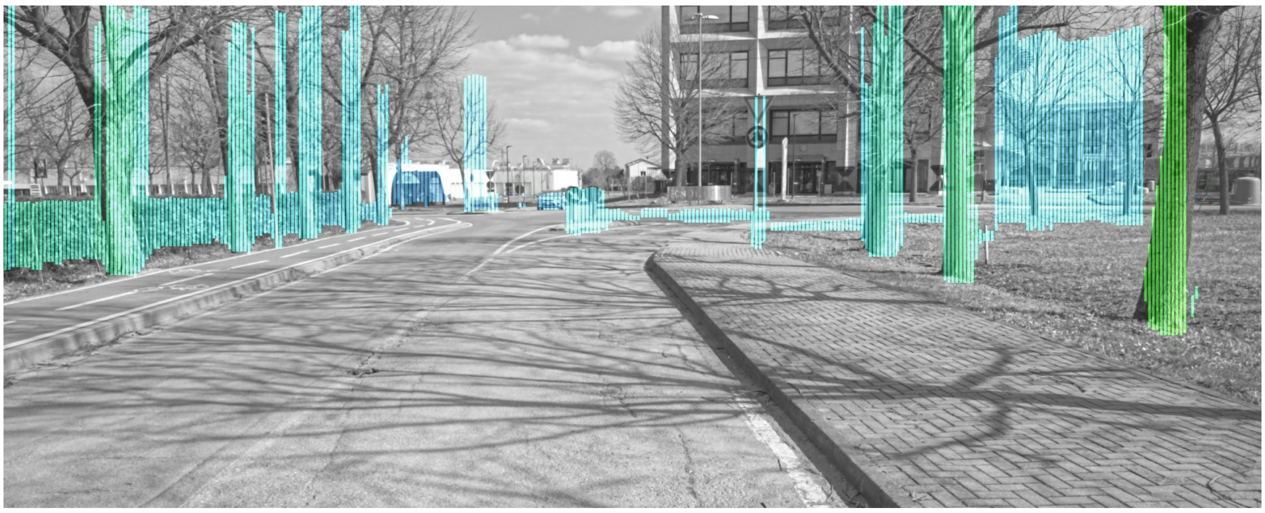
If any situation above is inevitable, setting it to 1.0 m would be better in practice.



DetObjectHeight = 0.3 m



DetObjectHeight = 0.4 m

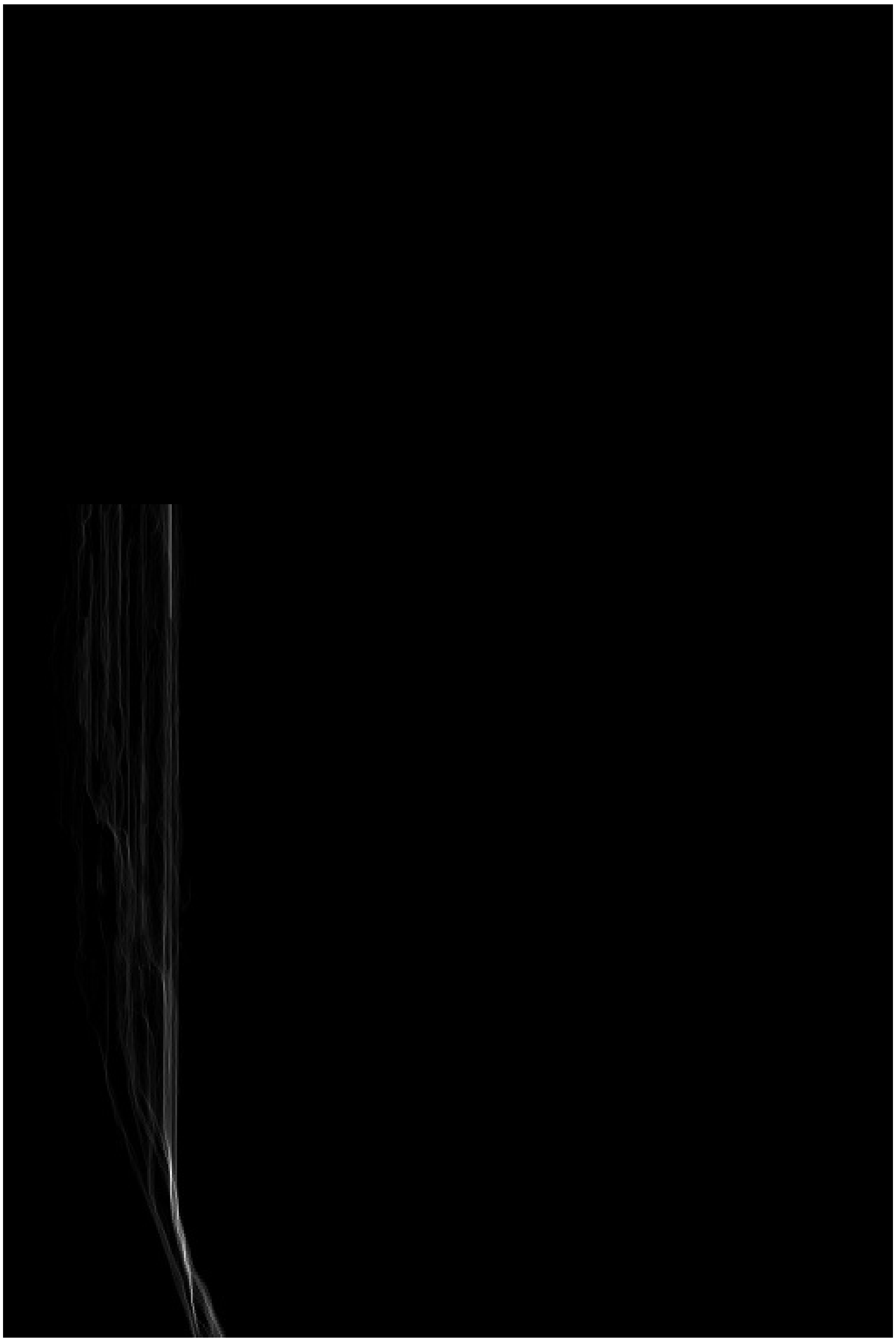


DetObjectHeight = 1.0 m

AMBA_CV_STIXEL_CAM_VDISP_PARAM_s

The sample code estimate extrinsic parameter based on v-disparity , which is not always useful when there is no sufficient area of ground to give it a clue.

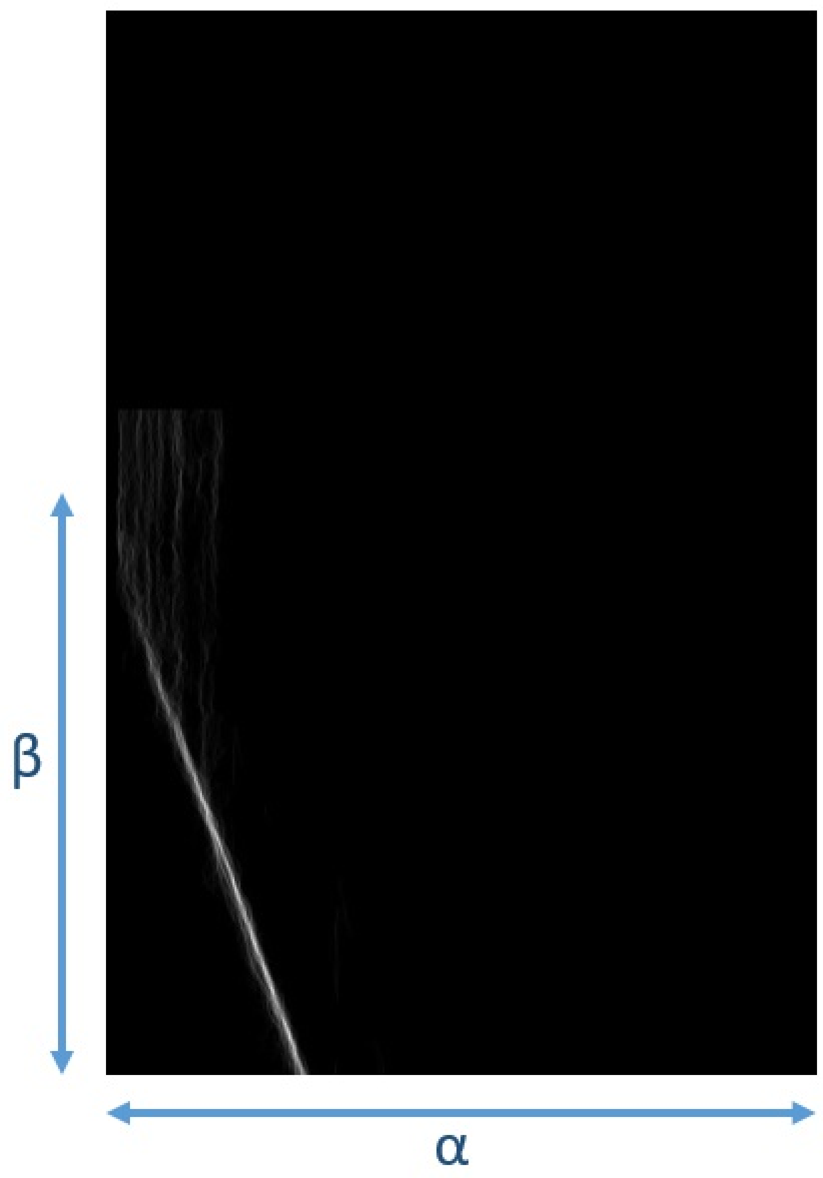
Quite small area of ground is visible



Derived v-disparity map, from which it's hard to infer extrinsic parameter

It uses reserved extrinsic parameter instead of inferring that when getting clueless v-disparity.

Following is the way to assess if v-disparity is qualified or not, both $XRatioThr$ and $YRatioThr$ are configurable and proportional to requirement of completeness of v-disparity.



$$QualifiedVdisp = (Y >= YRatioThr)?True : False$$

$$Y = \left(\sum_{v=BottomOfImage}^{v=BottomOfImage-\beta} X_v\right)/\beta$$

$$X_v = ((VdispMap(u_{estimate}, v)/\alpha) >= XRatioThr)?1 : 0$$

Other configurations

Following properties are hard coded in vpp file, rebuild is needed if anyone get modified

- Input dimension. (1920x768)
- Granularity of disparity. (8.4 format, e.g., 100 stand for 6.25 pixel)
- Reserved value of invalid disparity. (0xFFF)

Statistics

Here profile each step of the sample code

- Input disparity : (width, height) = (1920, 768)
- Output stixel width : 6
- Environment : CV2 with cv2_svc_cv_ut_defconfig

Profile

Step	Time(us)
VDisparity	2905
CameraPoseEstimation(ManualMode)	3412
PlaneEstimation	115
DisparityPreprocessing	491
FreeSpaceDetection	786
HeightSegmentation	2159
PackStixelInfo	5
Total	9873

Limitation And Suggestion

There are several limitation of this sample code, here list the commonest ones, and also give some corresponding suggestion.

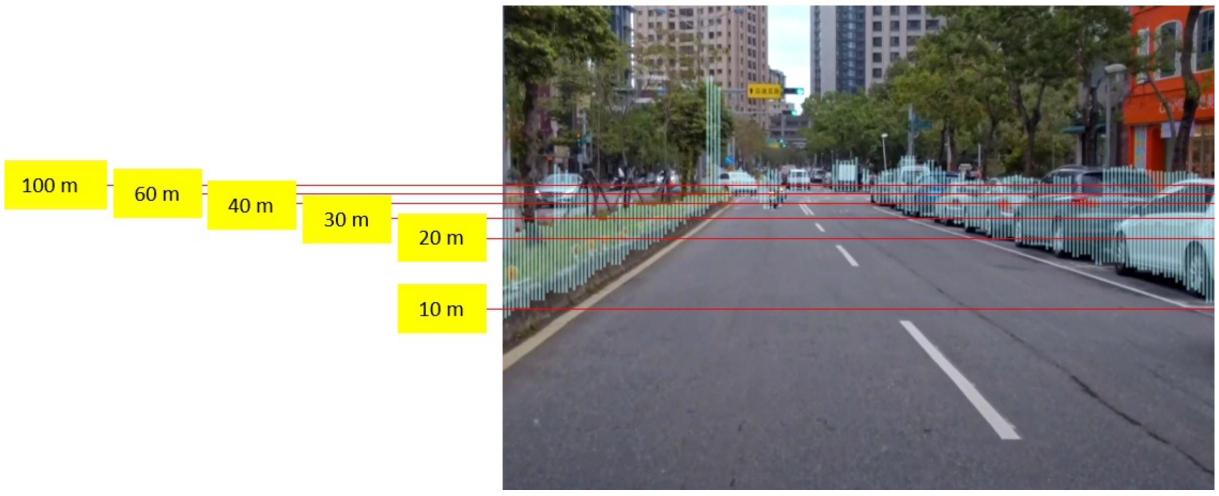
- In short, the largest distinguishable distance for this demo is ~= 100(meter), but object less than 60(meter) is more detectable.

  The farthest distance under the setting

  $(FocalLength, BaseLine, MinReliableDisparity) = (1590, 0.25, 4)$

  $Distance = (FocalLength * Baseline)/(MinReliableDisparity) \ = 99.375(meter)$

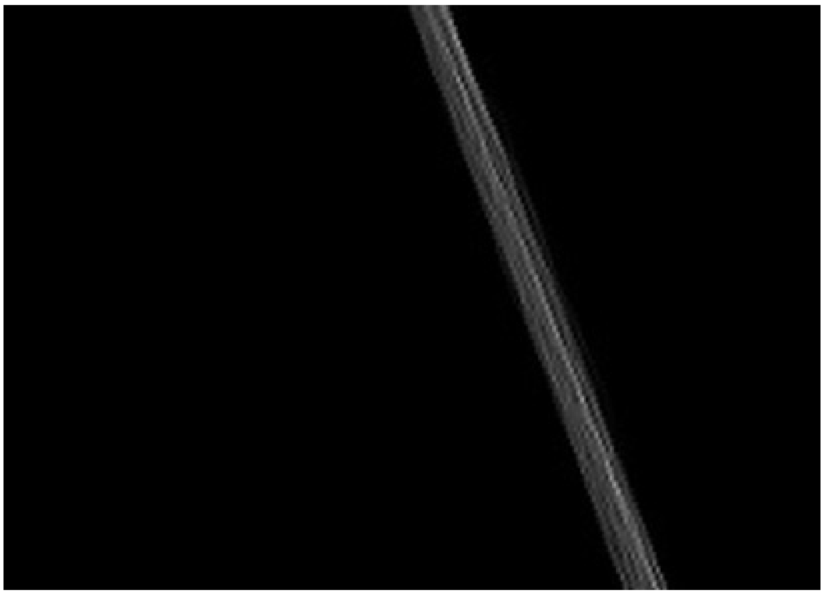
  But in practice the environment is noisy, and object locating within 60(meter) away can be observed almost the time

  [Suggestion] Install stereo camera with larger baseline and focal length if long range detection is in need.

- Using V-disparity to estimate ground plane is based on the assumption that the stereo head is well parallel to ground, but sometimes it's not the case. Once the stereo head is tilted, which often happens when the car is turning quickly, the line on the V-disparity will diverge so it can't use a line equation to represent ground plane.

  [Suggestion] Use other representation to model ground, such as curve surface or plane, or meshes ... etc.

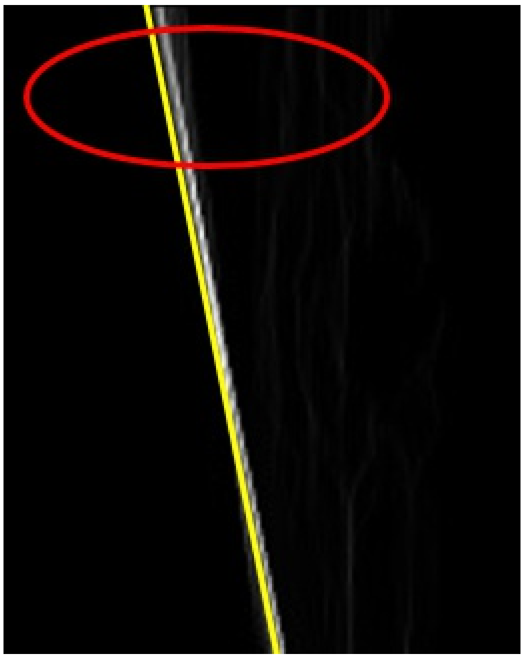


V-disparity of a tilted stereo head

- Road on near/far away side may appear to have different slope on V-disparity map, this happen when the facing road is going to be uphill.

  Using just 1 line equation will get bad estimation on far away side.

  [Suggestion] Use curve or piecewise straight line to git the v disparity instead of using only 1 straight line.



- In case image signal is noisy, which lead to generate noisy disparity value, it will make great impact on the accuracy of stixel.

  [Suggestion]

  1. Use distinct feature points, and do temporal matching to get ego-motion.
  2. Use spatial constraint to optimize position of stixels instead of treating each column independently. (Described in dynamic programing in ref [1])

Reference

1. H. Badino, U. Franke, and D. Pfeiffer, “The stixel world - a compact medium level representation of the 3d-world,” in DAGM Symposium, (Jena, Germany), September 2009.
2. D. Pfeiffer, U. Franke: “Efficient Representation of Traffic Scenes by means of Dynamic Stixels”, IEEE Intelligent Vehicles Symposium IV 2010, San Diego, CA, Juni 2010.
3. gishi523's implementation on stixel-world
4. J. Zhao, M. Whitty and J. Katupitiya, "Detection of non-flat ground surfaces using V-Disparity images," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 4584-4589, doi: 10.1109/IROS.2009.5354207.

Change Log

Version	Date	Log	Author
v1.0	05/07,2021	InitialDraft	HenryYu