Test of Bertozzi and Broggi's Inverse Perspective Mapping Functions

Based on "Stereo inverse perspective mapping: theory and applications" (Betozzi, Broggi, and Fasioli, 1998) along with "GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection" (Bertozzi and Broggi, 1998). Between these two papers, we were finally able to fill in the holes in the definitions of their equations, variable, and coordinates.

We modified variable names for convenience and reworked things slightly so that the world coordinate system would be right-handed and to allow for a non-square image.

Test Parameters

First set up camera and image parameters for a simple test case.

Image size: $m := 200$ rows (in the u direction)

$n := 300$ columns (in the v direction)

1/2 the camera viewing angle: $\alpha := 20\mathrm{deg}$

Camera location in world coordinates: $d_x := 1 \quad d_y := 2 \quad d_z := 1.5$

Camera angle with respect to forward: $\gamma_0 := 0\mathrm{deg}$

Camera angle with respect to horizon: $\theta_0 := 10\mathrm{deg}$

Transformation Equations

$$U(x,y) := \left(\frac{m-1}{2\cdot\alpha}\right)\cdot\left(\operatorname{atan}\left(\frac{d_z\cdot\sin\left(\operatorname{atan}\left(\frac{x-d_x}{y-d_y}\right)\right)}{x-d_x}\right)-\theta_0+\alpha\right)$$

(World to Image)

$$V(x,y) := \left(\frac{n-1}{2\cdot\alpha}\right)\cdot\left(\operatorname{atan}\left(\frac{x-d_x}{y-d_y}\right)-\gamma_0+\alpha\right)$$

$$X(u,v) := d_z\cdot\cot\left[\theta_0-\alpha+u\cdot\left(\frac{2\cdot\alpha}{m-1}\right)\right]\cdot\sin\left[\gamma_0-\alpha+v\cdot\left(\frac{2\cdot\alpha}{n-1}\right)\right]+d_x$$

(Image to World)

$$Y(u,v) := d_z\cdot\cot\left[\theta_0-\alpha+u\cdot\left(\frac{2\cdot\alpha}{m-1}\right)\right]\cdot\cos\left[\gamma_0-\alpha+v\cdot\left(\frac{2\cdot\alpha}{n-1}\right)\right]+d_y$$

Since we are assuming that the camera is viewing a perfectly flat plane, as the distance in front of the camera goes to infinity, there should be a distinct row in the image where the horizon appears.

$$\mathrm{uHorizon} := \lim_{y \to \infty} U(x,y) \to 49.750000000000000000$$ sure enough.

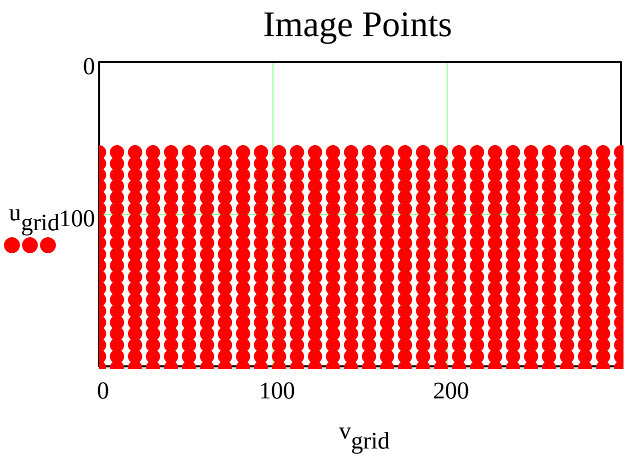
To test the functions, let's set up a grid of points in image coordinates and see where they map in world coordinates and vice versa.

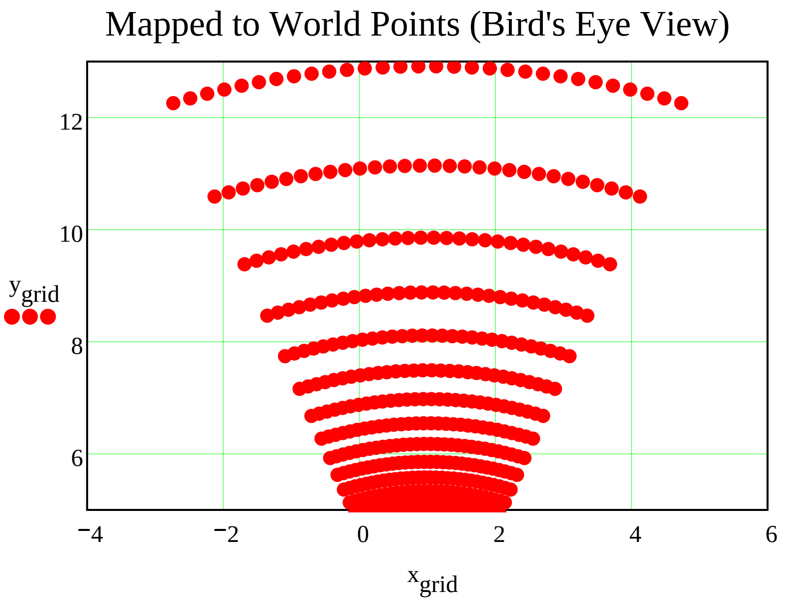
Image to World Grid Test

$\mathrm{uStart} := \mathrm{floor}(\mathrm{uHorizon}) + 10 \qquad k := 10$

$\mathrm{Xmesh} := \mathrm{CreateMesh}\left(X, \mathrm{uStart}, m, 0, n-1, \frac{m}{k}, \frac{n}{k}\right) \qquad \mathrm{Ymesh} := \mathrm{CreateMesh}\left(Y, \mathrm{uStart}, m, 0, n-1, \frac{m}{k}, \frac{n}{k}\right)$

$u_{grid} := \left(\mathrm{Xmesh}_1\right)_1 \qquad v_{grid} := \left(\mathrm{Xmesh}_1\right)_2 \qquad x_{grid} := \left(\mathrm{Xmesh}_1\right)_3 \qquad y_{grid} := \left(\mathrm{Ymesh}_1\right)_3$





World to Image Grid Test

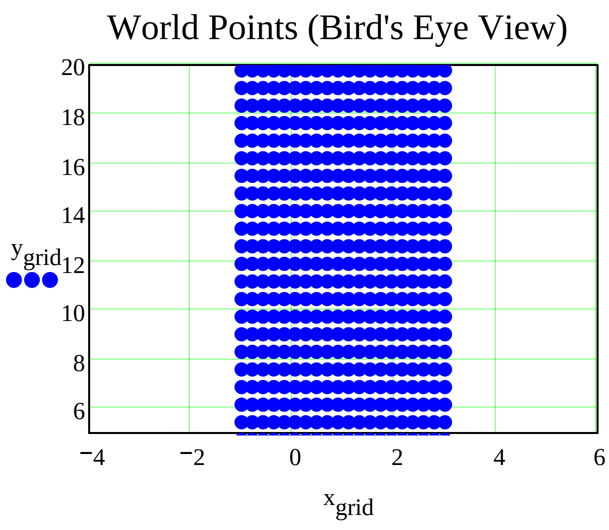
$xStart := -1$ $\quad$ $xEnd := 3$ $\quad$ $n_x := 20$

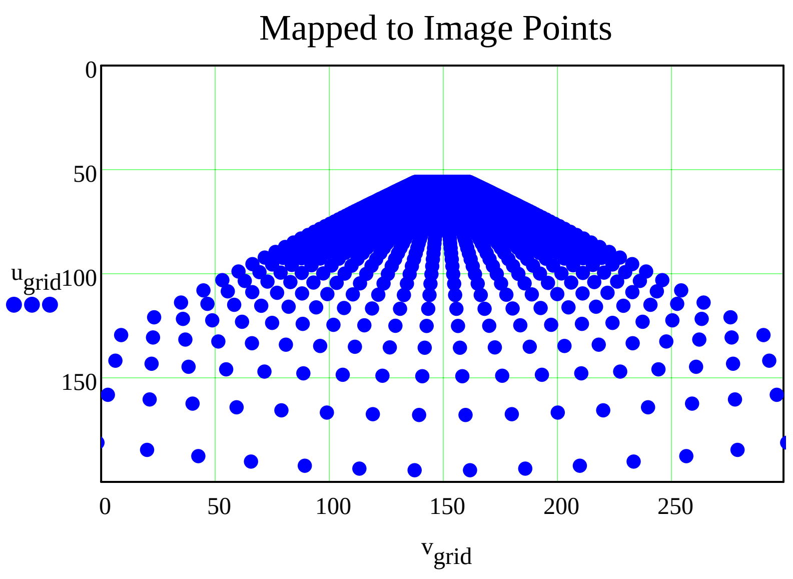
$yStart := 4$ $\quad$ $yEnd := 75$ $\quad$ $n_y := 100$

$Umesh := CreateMesh(U, xStart, xEnd, yStart, yEnd, n_x, n_y)$

$Vmesh := CreateMesh(V, xStart, xEnd, yStart, yEnd, n_x, n_y)$

$u_{grid} := \left(Umesh_1\right)_3$ $\quad$ $v_{grid} := \left(Vmesh_1\right)_3$ $\quad$ $x_{grid} := \left(Umesh_1\right)_1$ $\quad$ $y_{grid} := \left(Umesh_1\right)_2$





Comments on Grid Test Results

Looking at the transformation results, it is clear that while these equations get the general idea correct, there is some kind of error in their derivation because both mappings turn straight, horizontal lines into curves, which is not the correct result.

The transformations do get the general perspective effects correct. Image points map to world points which fan out in the x direction and spread out in the y direction, as expected. World points map to image points which get compacted due to the perspective effect, as expected. The only problem is the arcing that gets introduced, which could cause some real problems. This would make a straight stop line look like a curve making it more difficult to recognize.

In the defense of the equations, the amount of curvature introduced isn't too terrible and the authors presumably used these equations in their own vision applications and the issue didn't prevent them from obtaining good results. Also, our own test of the equations on a MATLAB generated road image gave reasonably good results. However, if we can figure out why the curvature is introduced and correct the problem, that would definitely be best since it would eliminate one more source of error.