3D Vehicle Reconstruction via Monocular Camera with Deep Learning Models and Direct Linear Transformation

2024-2 Robot Vision (M3228.003000)

Thomas Putzer
Computer Science and Engineering
Seoul National University
Seoul, South Korea
putzer_thomas@snu.ac.kr

Weihao Chao
Mechanical Engineering
Seoul National University
Seoul, South Korea
email address or ORCID

Anggraini Ditha
Civil and Environmental Engineering
Seoul National University
Seoul, South Korea
email address or ORCID

***Abstract*—Hello here goes the abstract**

I. Introduction

[1] (Leave this citation for now, otherwise it breaks (if we have 0 citations, and the command at the end))

Bla Bla Bla Bla

II. 3D Pose Estimation

($A = U\Sigma V^T$) [2] In the SVD $Ab = U\Sigma V^T b = 0$ implies $V^T b = 0$ since U is invertible and has no nontrivial solutions for $Ub = 0$. Thus $V^T b$ must be a vector where only the rows corresponding to zero singular values contribute to the null space. The columns of V that correspond to zero singular values in Σ combined form a basis for the null space of A.

$$\left[\begin{array}{ccc:c} & R & & t \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

$\leq$

```
#Pseudo code of the matching process
#cars keeps track of all the cars in the scene
#cars_n contains all the cars in the new frame
for old_car in cars:
    c, d = closest_car(cars_n, old_car)
    if d > max_distance:
        #the car left the frame
        cars.remove(old_car)
    else:
        old.match(c)
        #every car can only be matched once
        cars_n.remove(new_car)

#all remaining not matched new cars
for new_car in cars_n:
    #a new car has entered the frame
    cars.append(new_car)
```

double column text here. .double column text here. . . double column text here. . . double column text here. .

. double column text here. . .lkasj fundingasdf

lsakdfj laks ich icha heise laskdjf lasdffi cich ic ach .

lkasdjf lakjsd flkajsdl $x_p = (x'/2 + 0.5) \cdot width$ and $y_p = (y'/2 + 0.5) \cdot height$ fkjasdl fjasldj flaskdjf laskdjf laksjdfl kajsdflkajsdlfjasldkfjalsdkjflaskjdf laskdjf lasjkdfl aksjdflkasjd flaskjdf lkasdjf lkasjdf laksjdfl akjsdlfj asldkfj alsdjkf lasdjf laskdjf laskdjf laksdjfl aksdjfl askdjl askjf l

$$FOV = 2 \cdot \arctan\left(\frac{s_w}{2f}\right), r = n \cdot \tan\left(\frac{FOV}{2}\right) \tag{2}$$

double column text here. .double column text here. . . double column text here. . . double column text here. . . double column text here. . .lkasj fundingasdf double column text here. .double column text here. . . double column text here. . . double column text here. . . double column text here. . .lkasj fundingasdf double column text here. .double column text here. . . double column text here. . . double column text here. . . double column text here. . .lkasj fundingasdf

double column text here. .double column text here. . . double column text here. . . double column text here. . . double column text here. . .lkasj fundingasdf double column text here. .double column text here. . . double column text here. . . double column text here. . . double column text here. . .lkasj fundingasdf

Matrix representation of the standard projection onto the z = 1 plane.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z \end{bmatrix} \overset{\text{div z}}{\equiv} \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix} \tag{3}$$

The API projection Matrix

$$M_P = M_{P''}M_{P'} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & \frac{-(r+l)}{r-l} \\ 0 & \frac{2}{t-b} & 0 & \frac{-(t+b)}{t-b} \\ 0 & 0 & \frac{2}{f-n} & \frac{-(f+n)}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & c_3 & c_4 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1}$$

Fig. 1. The values r, l, t and b can be calculated from the focal length and sensor size of the camera (see 4). E.g., r can be calculated in the following way: $r = n \cdot \tan\left(\frac{FOV}{2}\right)$, $FOV = 2 \cdot \arctan\left(\frac{s_w}{2f}\right)$, ($s_w$ = sensor width, f = focal length, FOV = field of view).

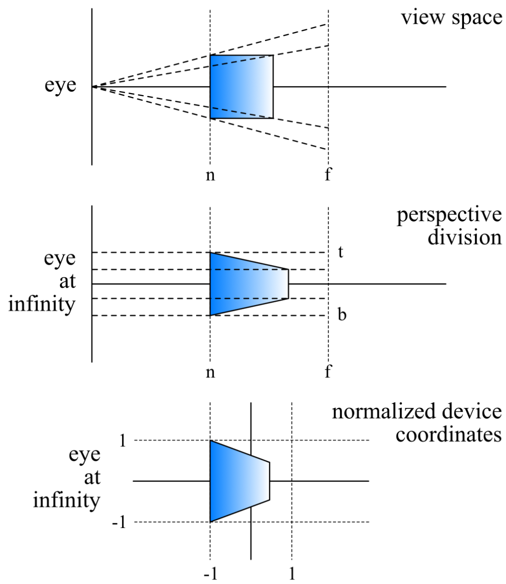

Fig. 2. Perspective projection from view space to NDC in 2d.

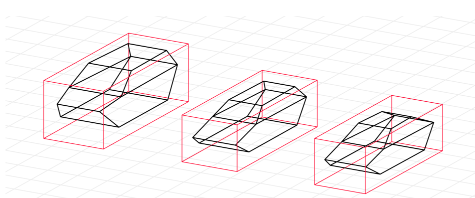

Fig. 3. Different car wireframes used. From left to right Ford Explorer, Volvo V60, Nissan Altima.

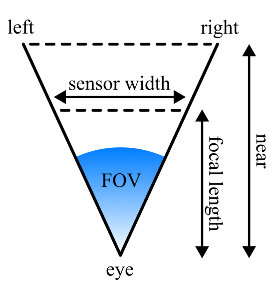

Fig. 4. The sensor width and the focal length are used to calculate the camera's field of view.

To get the screen-space position of a vertex, we multiply it by its Modelview Matrix and API Projection Matrix before doing the projective division. We have all the parameters except the Model Matrix. $p' = M_{API}M_{Modelview}p$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \overset{\text{div w}}{\equiv} \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & c_3 & c_4 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

If we multiply that out, we get for x' and y':

$$x' = \frac{c_1ax + c_1by + c_1cz + c_1d}{ix + jy + kz + l}$$
$$y' = \frac{c_2ex + c_2fy + c_2gz + c_2h}{ix + jy + kz + l}$$

We can multiply by the denuminator and solve for 0. Since this holds for every pair of points $(p_1, p_1'), (p_2, p_2'), ..., (p_n, p_n')$ that we track we can create the following system of equation: 4

References

[1] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, T. G. B. Team, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. A. Nowak, and E. L. Aiden, "Quantitative analysis of culture using millions of digitized books," *Science*, vol. 331, no. 6014, pp. 176–182, 2011. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1199644

[2] "Singular value decomposition," https://en.wikipedia.org/wiki/Singular_value_decomposition, accessed: 2024-12-06.

$$
\begin{bmatrix}
-c_1x_1 & -c_1y_1 & -c_1z_1 & -c_1 & 0 & 0 & 0 & 0 & x_1x_1' & y_1x_1' & z_1x_1' & x_1' \\
0 & 0 & 0 & 0 & -c_2x_1 & -c_2y_1 & -c_2z_1 & -c_2 & x_1y_1' & y_1y_1' & z_1y_1' & y_1' \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
-c_1x_n & -c_1y_n & -c_1z_n & -c_1 & 0 & 0 & 0 & 0 & x_nx_n' & y_nx_n' & z_nx_n' & x_n' \\
0 & 0 & 0 & 0 & -c_2x_n & -c_2y_n & -c_2z_n & -c_2 & x_ny_n' & y_ny_n' & z_ny_n' & y_n'
\end{bmatrix}
\begin{bmatrix} a \\ b \\ \vdots \\ k \\ l \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}
\tag{4}
$$