CS 3630 Lecture Notes on Computer Vision

Frank Dellaert

March 8, 2013

1 Image Formation

Image formation is, with $P^c = (X^c, Y^c, Z^c)$ expressed in the camera frame C:

$$x = X^c/Z^c$$

$$y = Y^c/Z^c$$

Converting to pixels, see Eq. 4.40 and 4.41 in book:

$$u = (k_u f)\, x + sy + u_0$$

$$v = (k_v f)\, y + v_0$$

where k_u and k_v are the density of pixels, measured in pixels/m. We can instead use the corresponding focal lengths α_u and α_v expressed in pixels:

$$u = \alpha_u x + sy + u_0$$

$$v = \alpha_v y + v_0$$

In general we express points in the world frame W, however, hence we need

$$\tilde{p} = A[I|0]P^w = A[I|0]T_w^c P^w$$

with A the calibration matrix

$$A = \begin{bmatrix} \alpha_u & s & u_0 \\ & \alpha_v & v_0 \\ & & 1 \end{bmatrix}$$

and $T_w^c \in SE(3)$ the transformation from camera to world

$$T_w^c = (T_c^w)^{-1} = \begin{bmatrix} R_c^w & t_c^w \\ 0 & 1 \end{bmatrix}^{-1}$$

Figure 1: Motion blur resulting from taking a picture from moving car, with the camera pointing in the direction of travel. Photo by Denim Dave, Flickr user, used under creative commons license.

2 Visual Servoing

Can we predict how things in the image will move given a certain camera motion? Certainly we have some intuitions, e.g., look at Figure 1: When the camera moves towards the scene at high speed, the resulting motion blur shows that the "optical flow" in the image flows from the middle of the image outwards, and the flow is much stronger (a) for nearby objects, and (b) towards the sides of the image. Similarly, the photo in image 2 illustrates that when we rotate the camera around the optical axis, the flow is circular around the image center. Can we quantify this relationship between camera motion and the apparent motion of scene points across the image plane? You bet.

If we move the camera from one world position T_{c1}^w to another T_{c2}^w using a twist $\dot{\xi} \triangleq \begin{bmatrix} \omega \\ v \end{bmatrix}$, we have

$$T_{c2}^w = T_{c1}^w \exp\left(\dot{\xi} t\right)$$

and the coordinates of a point P transform as

$$\begin{aligned} P^{c2} &= \left(T_{c2}^w\right)^{-1} P^w \\ &= \left(T_{c1}^w \exp\left(\dot{\xi} t\right)\right)^{-1} P^w \\ &= \exp\left(-\dot{\xi} t\right) \left(T_{c1}^w\right)^{-1} P^w \\ &= \exp\left(-\dot{\xi} t\right) P^{c1} \end{aligned}$$

Figure 2: Motion blur resulting from roll (rotation around the optical axis). Photo by Natesh Ramasamy, Flickr user, used under creative commons license.

For small t, e.g. δt, we have

$$\exp\left(-\dot{\xi}\delta t\right) \approx I - \begin{bmatrix} 0 & -\omega_z & \omega_y & v_x \\ \omega_z & 0 & -\omega_x & v_y \\ -\omega_y & \omega_x & 0 & v_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \delta t$$

and the point will transform as

$$\begin{bmatrix} X^{c2} \\ Y^{c2} \\ Z^{c2} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \omega_z\delta t & -\omega_y\delta t & -v_x\delta t \\ -\omega_z\delta t & 1 & \omega_x\delta t & -v_y\delta t \\ \omega_y\delta t & -\omega_x\delta t & 1 & -v_z\delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^{c1} \\ Y^{c1} \\ Z^{c1} \\ 1 \end{bmatrix}$$

from which we can compute

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} \omega_z Y - \omega_y Z - v_x \\ \omega_x Z - \omega_z X - v_y \\ \omega_y X - \omega_x Y - v_z \end{bmatrix}$$

where I omitted the coordinate frame superscripts for simplicity, but this is in the camera frame. The linear velocity part is easy to understand: if we move the camera with velocity v, a world point will appear to move with the opposite velocity in the camera frame. The rotation is analogous, but the motion of the point is bigger the further the point is from the camera origin.

We can then ask what happens to the projection (x, y) of that point? By applying the normal rules of differentiation, we have

$$\dot{x} = \frac{\partial X/Z}{\partial t} = \frac{\dot{X}Z - X\dot{Z}}{Z^2} = \frac{1}{Z^2}Z(\omega_z Y - \omega_y Z - v_x) - \frac{1}{Z^2}X(\omega_y X - \omega_x Y - v_z)$$
$$= (xy)\omega_x - (1+x^2)\omega_y + y\omega_z - \frac{1}{Z}v_x + \frac{x}{Z}v_z$$

A similar calculation for y yields

$$\dot{y} = (1+y^2)\omega_x - (xy)\omega_y - x\omega_z - \frac{1}{Z}v_y + \frac{y}{Z}v_z$$

Hence, for each point, we have the following linear mapping between spatial twist $\dot{\xi}$ and the change in projection:

$$\dot{p} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} xy & -(1+x^2) & y & -1/Z & & x/Z \\ (1+y^2) & -xy & -x & & -1/Z & y/Z \end{bmatrix} \begin{bmatrix} \omega \\ v \end{bmatrix} = J(x, y, Z)\dot{\xi}$$

where $J(x, y, Z)$ is a Jacobian that depends on both the position (x, y) of the point in the image, and on the depth Z of the point.

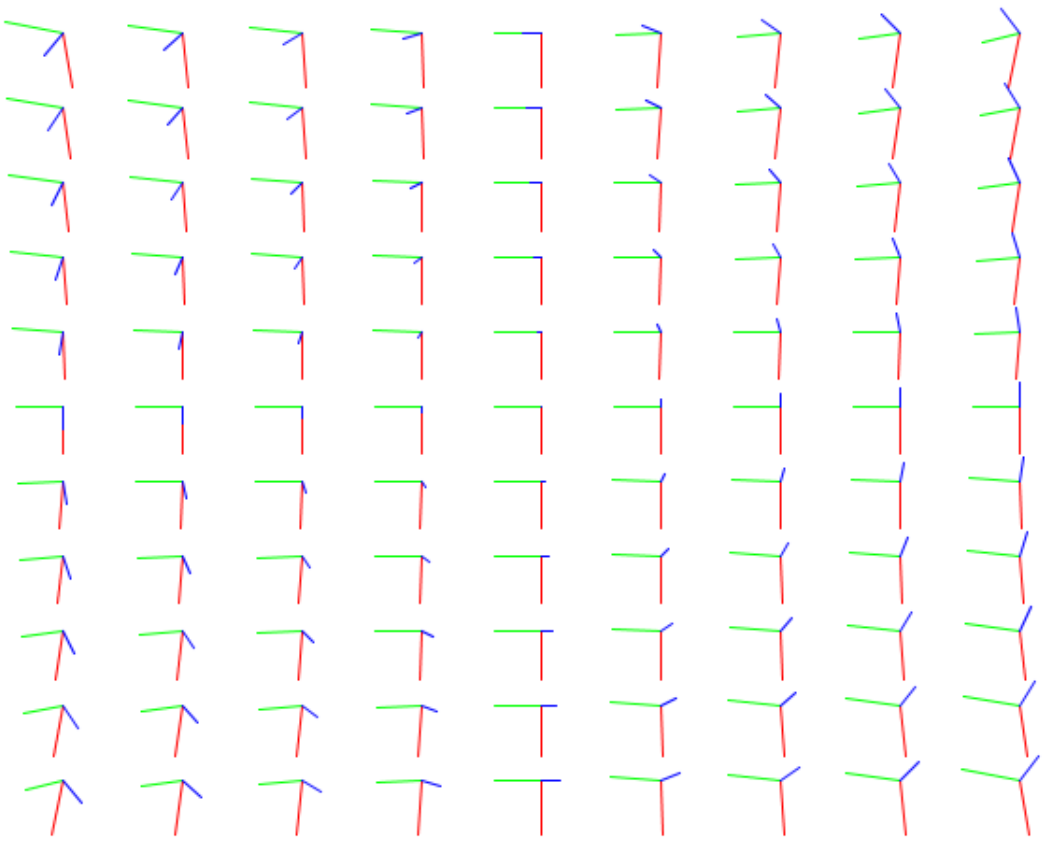

Figure 3: Angular velocity part of the Jacobian, evaluated at a regular grid.

In Figure 3 the angular velocity part of the Jacobian is visualized as red, green, and blue vectors for rotation around the camera's X,Y, and Z axes, respectively. I evaluated the Jacobian on a regular grid in the image, and it is clear that the Jacobian depends on the location in the image. To get some intuition, look at the red vectors only: in response to a CCW rotation around X (camera pitches up), all the projected points in the image move down. Try it with a camera, and note that **the amount of movement in response to a pure rotation does not depend on the distance to the scene points.** A CW rotation around Y corresponds to camera yaw to the right, and projections move opposite, to the left. Finally a CCW rotation around Z corresponds to clockwise camera roll, and projections move counter-clock-wise in response. Note that the amount of movement is zero at the center and increases towards the sides of the image, but again, does not depend on depth.

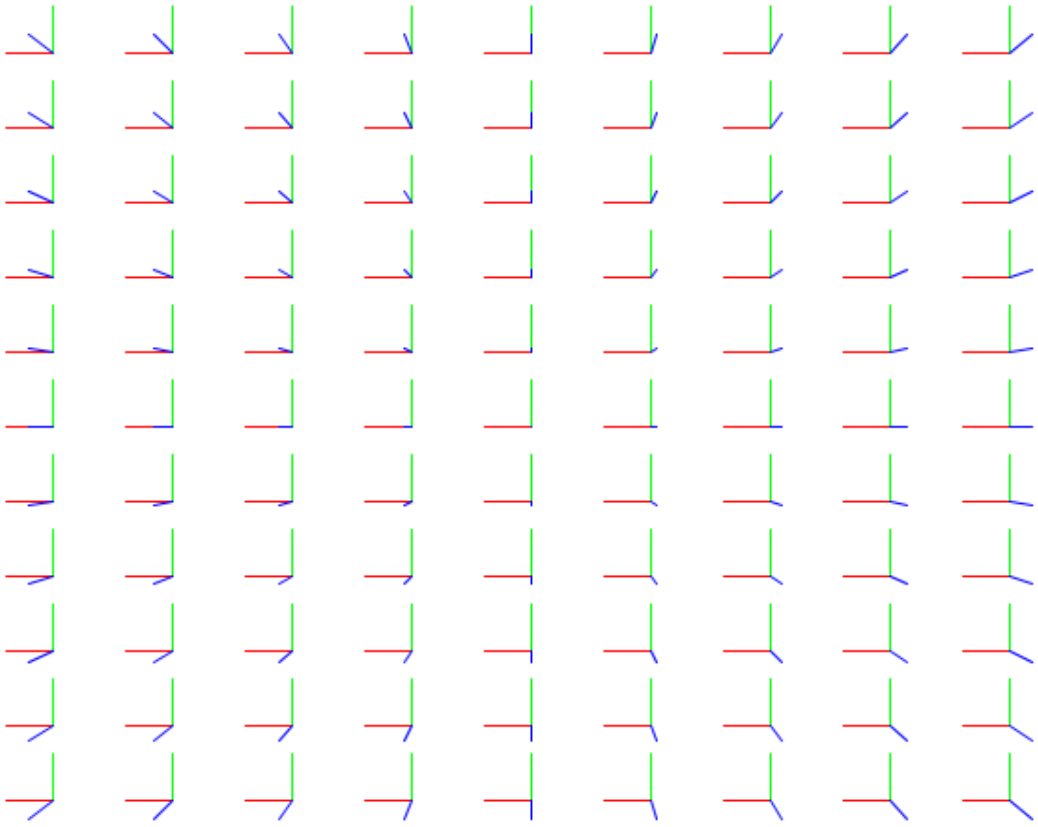

Figure 4: Linear velocity part of the Jacobian, evaluated at a regular grid, using $Z = 1$.

The linear velocity part of the Jacobian is shown in Figure 4. Again, the red vectors correspond to a positive velocity of the camera along its X-axis, and we see that all points in the image move in the opposite direction. The same holds for a positive velocity in Y, which is down, and hence green vectors point up. The blue vectors are the most interesting: in response to a forward motion of the camera, all points projected in the camera will move outwards from the center!