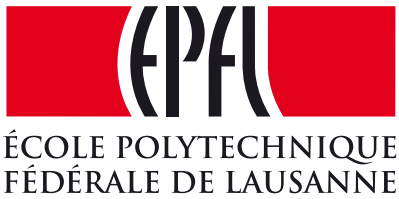

Visual Terrain Estimation For Legged Robots

Frederike Dümbgen

Master's Thesis
Ecole polytechnique fédérale de Lausanne
Department of Mechanical Engineering
Supervisors: Alireza Karimi (EPFL)
Michael Blösch (ETHZ)
Philipp Krüsi (ETHZ)
Dominik Schindler (ETHZ)

Zurich, 29 July 2016

1

Notation and Basics

1.1 Geometry

1.1.1 Vectors and Coordinate Frames

The vector and coordinate frame notations used throughout this report follow the conventions introduced in [**?**]. If $\mathcal{M}$ is the coordinate frame associated with the map, and A, B are two 3D points, then ${}_A\boldsymbol{r}_{AB}$ contains the coefficients of the vector pointing from A to B expressed in the map coordinate system. In the same manner, the rotation matrix $\boldsymbol{C}_{BA}$ is the (passive) matrix linking the coordinates in a frame $\mathcal{A}$ to the coorindates in $\mathcal{B}$ like

$$ {}_B\boldsymbol{r}_{AB} = \boldsymbol{C}_{BA}\,({}_A\boldsymbol{r}_{AB}) \ . \tag{1.1} $$

1.1.2 Rotation Representations

For convenient handling of rotations, i.e. their derivatives and updates, the rotation vectors $\boldsymbol{\Phi}_{C_kM}$, $\boldsymbol{\Phi}_{SM} \in \mathbb{R}^3$ are introduced as recommended in [**?**] to represent the relative orientations of the stereo and camera frames with respect to the map frame, respectively. The corresponding updates in rotation are denoted by $\boldsymbol{\varphi}_{C_kM}$and $\boldsymbol{\varphi}_{SM}$.

The new rotation $\boldsymbol{\Phi}^{k+1}_{C_kM}$, resulting from a relatively small change in rotation $\boldsymbol{\varphi}_{C_kM}$, can be conveniently expressed using the $\boxplus$-operator as

$$ \begin{aligned} \boldsymbol{\Phi}^{k+1}_{C_kM} &= \boldsymbol{\Phi}^{k}_{C_kM} \boxplus \boldsymbol{\varphi}_{C_kM} && (1.2)\\ &= \exp(\boldsymbol{\varphi}_{C_kM}) \circ \boldsymbol{\Phi}^{k}_{C_kM} && (1.3)\\ &= \boldsymbol{C}(\boldsymbol{\varphi}_{C_kM})\boldsymbol{C}(\boldsymbol{\Phi}^{k}_{C_kM}) && (1.4) \end{aligned} $$

The mapping $\boldsymbol{C} : SO(3) \implies \mathbb{R}^{3\times3}$ applied to the orientation vector $\boldsymbol{\Phi}_{C_kM}$corresponds

to the rotation matrix $\boldsymbol{C}_{C_kM}$since it is defined as

$$\boldsymbol{\Phi}_{C_kM}(\boldsymbol{r}) := \boldsymbol{C}(\boldsymbol{\Phi}_{C_kM})\boldsymbol{r} \tag{1.5}$$

$$= \boldsymbol{C}_{C_kM}(\boldsymbol{r}) \ . \tag{1.6}$$

The transformation defined in 1.6 can be derived with respect to the orientation vector $\boldsymbol{\Phi}_{C_kM}$and the vector $\boldsymbol{r}$, yielding

$$\frac{\partial \boldsymbol{\Phi}_{C_kM}(\boldsymbol{r})}{\partial \boldsymbol{r}} = \boldsymbol{C}(\boldsymbol{\Phi}_{C_kM}) \ , \quad \frac{\partial \boldsymbol{\Phi}_{C_kM}(\boldsymbol{r})}{\partial \boldsymbol{\Phi}} = -(\boldsymbol{\Phi}_{C_kM}(\boldsymbol{r}))^{\times} \ , \tag{1.7}$$

and

$$\frac{\partial (\boldsymbol{\Phi}_{C_kM} \circ \boldsymbol{\Phi}_{SM})(\boldsymbol{r})}{\partial \boldsymbol{\Phi}_{SM}} = \boldsymbol{C}(\boldsymbol{\Phi}_{C_kM}) \ . \tag{1.8}$$

1.2 Camera

1.2.1 Camera Model

Having introduced the orientation vectors, the camera poses can be described by

$$\boldsymbol{\xi}_{C_k} := \begin{bmatrix} {}_M\boldsymbol{r}_{MC_k} & \boldsymbol{\varphi}_{C_kM} \end{bmatrix}^T$$

$$\boldsymbol{\xi}_S := \begin{bmatrix} {}_M\boldsymbol{r}_{MS} & \boldsymbol{\varphi}_{SM} \end{bmatrix}^T \ .$$

The transformation from object space coordinates ${}_M\boldsymbol{r}_{MX_j}$to pixel coordinates $\boldsymbol{u}_{k,j}$ can be split up in the transformation

$${}_{C_k}\boldsymbol{r}_{C_kX_j} = \boldsymbol{C}_{C_kM}({}_M\boldsymbol{r}_{MX_j} - {}_M\,\boldsymbol{r}_{MC_k}),$$

which is essentially a coordinate transformation from the map frame to the camera frame using the camera's extrinsic parameters, and

$$\boldsymbol{u}_{k,j} = \boldsymbol{K}_k D_k(\boldsymbol{\pi}({}_{C_k}\boldsymbol{r}_{C_kX_j})), \tag{1.9}$$

which assigns a pixel value to each camera coordinate using the camera model and its intrinsic parameters.

The model used to approximate the real camera is a pinhole model suspect to radial - tangential distortion. A pinhole model assigns to each point ${}_{C_k}\boldsymbol{r}_{C_kX_j}$its projection

$$\tilde{\boldsymbol{u}} = \boldsymbol{\pi}({}_{C_k}\boldsymbol{r}_{C_k X_j}) = \begin{bmatrix} x/z & y/z & 1 \end{bmatrix}^T \tag{1.10}$$

The obtained normalized coordinates $\tilde{\boldsymbol{u}}$ are then distorted using the distortion model D_k defined by [**?**], which can be scaled and shifted using the camera matrix $\boldsymbol{K}_k$ in order to give pixels the final pixels $\begin{bmatrix} u, v \end{bmatrix}$ lying in $\begin{bmatrix} 0 \dots w, 0 \dots h \end{bmatrix}$.

1.2.2 Stereo Parameters

The same considerations as above can be made when unrectified instead of rectified images are available. The camera calibration in general provides us with information on the orientations of the cameras with respect to the rectified image plane $\boldsymbol{R}_k$ and their baseline b. [**?**] [**?**, p. 523f] In the present notation, this means in particular that

$$\boldsymbol{C}_{SC_k} = \boldsymbol{R}_k \quad \text{and} \quad {}_S\boldsymbol{r}_{C_1 C_2} = b \tag{1.11}$$

are known. The distortion coefficients D_k are zero and the obtained normalized points need to be rotated to lie in the same rectified image plane. The camera projection 1.9 for the rectified case therefore becomes

$$\tilde{\boldsymbol{w}}_{k,j} = \boldsymbol{K}'_k \boldsymbol{R}_k(\boldsymbol{\pi}({}_{C_k}\boldsymbol{r}_{C_k X_j})),$$

where $\boldsymbol{K}'_k$ is a modified camera matrix which, depending on the camera calibration, can be used to scale the image to void non-defined image borders resulting from image undistortion and rectification.

For a given set of rectified and unrectified images of a well calibrated camera, there are strong correspondances between the pixels obtaiend from projections of 3D points to the rectified and the unrectified images respectively. This connection was exploited to test the correctness of the above transformations and the provided camera paramteres, as shown in Appendix **??**.

1.2.3 Interpolation

The image has to be interpolated around the obtained floating-precision pixel locations using any common interpolation kernel $h(i, j)$ and summing over

$$I(x,y) = \sum_{i=1}^{s}\sum_{j=1}^{s} I(x_i, y_j)h(i,j) \tag{1.12}$$

$$= \sum_{i=1}^{s}\sum_{j=1}^{s} I(x_i, y_j)h(i)h(j) \tag{1.13}$$

The indices x_i and y_i correspond to the location of the image where the interpolation function $h(i,j)$ takes its maximum value.

The applied scheme in this project was the Keys cubic convolution interpolation kernel [**?**], defined by

$$h(x) = \begin{cases} 0.5(x-2)^3 + 2.5(x-2)^2 + 4(x-2) + 2 & 0 \leq x < 1 \\ -1.5(x-2)^3 - 2.5(x-2)^2 + 1 & 1 \leq x < 2 \\ 1.5(x-2)^3 - 2.5(x-2)^2 + 1 & 2 \leq x < 3 \\ -0.5(x-2)^3 + 2.5(x-2)^2 - 4(x-2) + 2.0 & 3 \leq x < 4 \end{cases} \tag{1.14}$$

Other choices of interpolation schemes, such as bilinear or quadratic interpolation kernels, could have been considered but no further investigation was done in this direction since the interpolation scheme did not show any significant effect on the results.

1.2.4 Disparity Images

In order to quantify the error between the obtained 3D map and the reference map, disparity images of both maps are created and their difference is evaluated. This approach is preferred over other approaches such as the sum over all height differences, since at sharp edges a little lateral offset can induce very high differences, but the disparity map's error stays relatively constrained, depending on the resolution used. The disparity map also offers a welcome intermediate visualization that gives a very intuitive first impression of the accuracy of the obtained point cloud. Finally, the disparity being simply defined as $d = bf_x/z$, with z the depth of the point in the camera frame $\mathcal{C}$, a disparity map can be very easily calculated, as shown in Algorithm **??**).

It can not be assured that every pixel in the obtained disparity map has an assigned disparity after running Algorithm 1. By reducing the resolution of the disparity image by a sufficiently high factor with respect to the original images, however, the number of non-defined pixels can be limited considerably. In the present case, a good choice of this factor was shown to be 2-4 and the few non-defined pixels left were assigned the average value of the surrounding non-zero pixels, leading to smooth disparity images.

Algorithm 1: Create a disparity map from 3D coordinates

Input : Reduced camera projection Proj and map coordinates $\boldsymbol{x}$
Output: Disparity map

```
Disparities[1...h, 1...w];
Counter[1...h, 1...w];
for i = 1 to N do
    [u, v, z] = Proj(x(i)) ;
    d = b f_x / z ;
    Disparities[v, u] += d;
    Counter[v, u] ++;
end
return Disparities/ Counter;
```

1.3 Splines

What are splines? Why do we use them? Why are they so great? bla bla bla bla.

1.3.1 Interpolation

The naming and indexing conventions for the splines formulations are inspired by [**?**].

$$h(x) = \begin{cases} 1/6x^3 & 0 \leq x < 1 \\ -1/2x^3 + 2x^2 - 2x + 2/3 & 1 \leq x < 2 \\ 1/2x^3 - 4x^2 + 10x - 22/3 & 2 \leq x < 3 \\ -1/6x^3 + 2x^2 - 8x + 32/3 & 3 \leq x < 4 \end{cases} \tag{1.15}$$

$$_M\boldsymbol{r}_{MX_j} = \Big[x_j, y_j, h(x_j, y_j)(\boldsymbol{a})\Big]^T$$

$$\begin{aligned} h(x_j) &= \sum_{i=0}^{M} a_i h_i(x_j) \\ &= \sum_{i=k}^{k+s} a_i h_i(x_j), \text{ for } j = 1 \dots N \end{aligned}$$

1.3.2 Bending energy

Formal definition and numerical implementation?

$$B(h) = \iint_{\mathbb{R}^2} \left(\frac{\partial^2 h}{\partial x^2}(x,y)\right)^2 + \left(\frac{\partial^2 h}{\partial x \partial y}(x,y)\right)^2 + \left(\frac{\partial^2 h}{\partial y^2}(x,y)\right)^2 \mathrm{d}x\mathrm{d}y \tag{1.16}$$

1.3.3 Gradient energy

Formal definition and numerical implementation?

$$G(h) = \iint_{\mathbb{R}^2} \left(\frac{\partial h}{\partial x}(x,y)\right)^2 + \left(\frac{\partial h}{\partial y}(x,y)\right)^2 \mathrm{d}x\mathrm{d}y \tag{1.17}$$