Camera calibration

DEPARTMENT OF ELECTRONICS AND ELECTRICAL COMMUNICATION ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

Joshua Peter Ebenezer

July 23, 2018

Abstract

This technical report aims to give a brief overview of camera calibration using Zhang's method, and estimating the intrinsic and extrinsic parameters of a camera from it. This report is a compilation of insights gained regarding calibration over the summer of 2017.

Contents

1 Introduction **3**

2 Background **3**
2.1 The pinhole camera model 3
2.2 The intrinsic camera matrix 4
2.3 The extrinsic camera matrix 5

3 Construction of equations **5**
3.1 The homography between the model plane and the image 5
3.2 Constraints on the intrinsic parameters 6

4 Solving the equations **6**
4.1 Initial estimate 6
4.2 Maximum likelihood estimation 7

Appendices **7**

A Eigenvalues, Eigenvectors and SVD **7**

B Least squares method **8**

C Levenberg-Marquardt algorithm **9**

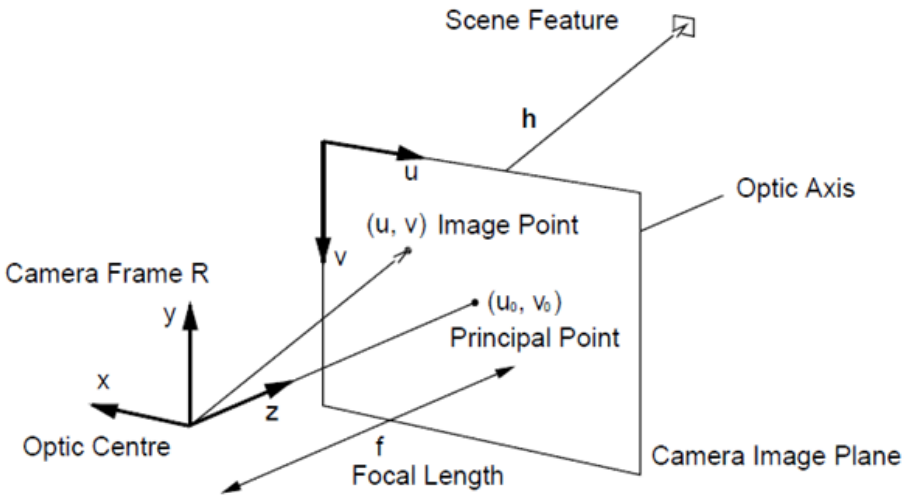

Figure 2.1: Pinhole camera model

1 Introduction

Camera calibration is necessary in computer vision to extract 3D information from 2D images. This information, which includes, but may not be limited to, the depth of each scene point from the camera, the location of the camera in world co-ordinates and the focal length of the camera, can then be used in a variety of tasks. Camera calibration also enables the correction of images that suffer from optical distortion.

2 Background

2.1 The pinhole camera model

3D scene points in the real world are projected into a 2D homogeneous image by the camera, and this is modeled using an ideal pinhole camera. The mapping is called a perspective projection. Figure 2.1 shows the model and the mapping:

In an actual pinhole camera, the image is inverted w.r.t the actual object. For simplicity, we will mirror the image plane across the focal point and deal with a virtual image plane instead, where the image is upright. The camera's principal axis is the line that passes through the focal point (or the optical centre) and the image plane, and its intersection with the image plane is called the principal point.

Say a 3D point is modeled as $[X, Y, Z]^T$, and it's corresponding mapping in the image plane is represented by pixel location $[u, v]^T$. From the model, it is seen that applying the principal of similar triangles,

$$u = u_0 + \frac{f_x X}{Z} \tag{2.1}$$

$$v = v_0 + \frac{f_y Y}{Z} \tag{2.2}$$

where f_x and f_y are the focal length of the camera divided by the pixel dimensions in the x and y directions respectively.

To avoid this non-linear transformation, we add another element to the system to make it homogeneous. We define the augmented vectors $\vec{m} = [u, v, 1]^T$ and $\vec{w} = [X, Y, Z, 1]^T$. Assuming that the camera frame and the world frame are identical, we can represent the pinhole camera model by the following matrix equation.

$$s\vec{m} = \mathbf{A}\vec{w} \tag{2.3}$$

where **A** is called the intrinsic camera matrix, defined in the following section.

2.2 The intrinsic camera matrix

The intrinsic camera matrix is a matrix **A**, that encapsulates the information pertaining to the internal, or intrinsic, parameters of a camera, including the focal length, the dimensions of pixels in the image produced by the camera, and the origin of the pixel grid system in the camera image. It is also referred to as the camera calibration matrix in some texts. **A** is defined as:

$$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

The quantities α, β, γ, u_0 and v_0 are defined in the following manner: α is the focal length of the camera, F, in world units, divided by the length of each pixel in the x direction, d_x, also in world units. This is mathematically expressed as:

$$\alpha = f_x = \frac{F}{d_x} \tag{2.4}$$

β is similarly defined as the focal length of the camera divided by the length of each pixel in the y direction.

$$\beta = f_y = \frac{F}{d_y} \tag{2.5}$$

γ describes the skew of the x and y image axes, or in other words the skew of each pixel. Pixels may not always be perfect squares. Analog video systems like NTSC define non-square pixels. Inaccurate synchronization of the pixel sampling process by images acquired by frame grabbers may also cause skewness.

u_0 and v_0 are the co-ordinates of the principal point relative to the origin of the image. The location of the origin of the image depends on the convention used. It may be at the top left corner of the image, as is used generally in image processing, or it may be at the centre of the image.

The preceding treatment assumed that the camera frame and the world frame are the same, but this may not always be the case. The camera frame can be seen as an affine transformation of the world frame. The following section deals with this.

2.3 The extrinsic camera matrix

The camera frame can be seen as an affine transformation of the world frame. i.e. rotated by **R**, where **R** is a 3 x 3 matrix that encapsulates rotation about the three axes, and translated by **t**, where **t** is a 3 x 1 column vector that elements that represent translations in the x, y and z directions. Rotations about the x, y and z axes, by angles ϕ, θ, and ψ respectively, can be represented by the following three matrices.

$$\mathbf{R_x}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix}$$

$$\mathbf{R_y}(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

$$\mathbf{R_z}(\psi) = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

R is the product of these matrices, and thus has three degrees of freedom.

$$\mathbf{R} = \mathbf{R_x}\mathbf{R_y}\mathbf{R_z} \tag{2.6}$$

t is represented as

$$\mathbf{t} = [t_x \ t_y \ t_z]^T \tag{2.7}$$

The intrinsic and extrinsic matrices are used to represent the perspective projection as

$$s\vec{m} = \mathbf{A}\,[\mathbf{R} \mid \mathbf{t}]\vec{w} \tag{2.8}$$

where s is a scale factor.

Defining the camera matrix, **P**, as

$$\mathbf{P} = \mathbf{A}\,[\mathbf{R} \mid \mathbf{t}] \tag{2.9}$$

(2.8) can be written as

$$s\vec{m} = \mathbf{P}\vec{w} \tag{2.10}$$

R is a 3 x 3 matrix with 3 degrees of freedom, **t** is a 3 x 1 column vector with 3 degrees of freedom, and **A** is a 3 x 3 matrix with 5 degrees of freedom. Thus, **P** is a 3 x 4 matrix with 11 degrees of freedom.

3 Construction of equations

3.1 The homography between the model plane and the image

P has 12 entries and 11 degrees of freedom. Hence, without loss of generality, we can assume that the model plane is on Z=0 in the world coordinate system or frame.

Representing the columns of **R** as $\mathbf{r_1}$, $\mathbf{r_2}$ and $\mathbf{r_3}$, (2.9) can be written as

$$\begin{aligned} s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \mathbf{A}\,[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{r_3}\ \mathbf{t}]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \\ &= \mathbf{A}\,[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{t}]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \end{aligned} \tag{3.1}$$

We will continue to use $\overrightarrow{w}$ to represent the world coordinates of the scene point, but with Z=0 we will now use $\overrightarrow{w} = [X\ Y\ 1]^T$. We can now represent the perspective projection by a homography **H**.

$$s\overrightarrow{m} = \mathbf{H}\overrightarrow{w} \tag{3.2}$$

with

$$\mathbf{H} = [\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{t}] \tag{3.3}$$

H has 8 degrees of freedom as it is a homography.

3.2 Constraints on the intrinsic parameters

If we denote the columns of the homography matrix **H** as $\mathbf{h_1}$, $\mathbf{h_2}$, and $\mathbf{h_3}$, then

$$[\mathbf{h_1}\ \ \mathbf{h_2}\ \ \mathbf{h_3}\ ] = \lambda\,\mathbf{A}\,[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{t}] \tag{3.4}$$

where λ is an arbitrary scalar. $\mathbf{r_1}$ and $\mathbf{r_2}$ are orthonormal, i.e. $\mathbf{r_1}.\mathbf{r_2}$=0 and $\|\mathbf{r_1}\| = \|\mathbf{r_2}\| = 1$, since they are columns of the rotation matrix. Hence,

$$\mathbf{h_1}^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h_2} = 0 \tag{3.5}$$

$$\mathbf{h_1}^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h_1} = \mathbf{h_2}^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h_2} = 1 \tag{3.6}$$

4 Solving the equations

4.1 Initial estimate

Let

$$\mathbf{B} = \mathbf{A}^{-T}\mathbf{A}^{-1} \tag{4.1}$$

B is symmetric, because **A** is upper triangular. So let **B** be defined by a 6D vector $\overrightarrow{b} = [B_{11}, B_{12}, B_{13}, B_{22}, B_{23}, B_{33}]^T$. Let the ith column of **h** be $\mathbf{h_i}$. Then we define $\mathbf{v_{ij}}$ by the relation

$$\mathbf{h_i}^T\mathbf{B}\mathbf{h_j} = \mathbf{v_{ij}}^T\mathbf{b} \tag{4.2}$$

where

$$\begin{aligned} \mathbf{v}_{ij} = [&h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i1}h_{j3} + h_{i3}h_{j1}, \\ &h_{i2}h_{j2}, h_{i2}h_{j3} + h_{i3}h_{j2}, h_{i3}h_{j3}]^T \end{aligned} \tag{4.3}$$

(3.5) and (3.6) can now be written as

$$\mathbf{v}_{12}^T\mathbf{b} = \mathbf{v}_{21}^T\mathbf{b} = 0 \tag{4.4}$$

$$\mathbf{v}_{11}^T\mathbf{b} = \mathbf{v}_{22}^T\mathbf{b} \tag{4.5}$$

This is written compactly as

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = 0 \tag{4.6}$$

b is constant for a given camera and only depends on the intrinsic parameters of it. $\mathbf{v}_{ij}$ depends on the homography matrix and varies for different views of the model planes. So if n views of the model plane are taken, then $2n$ equations can be generated and stacked in (3.12), forming

$$\mathbf{V}\mathbf{b} = 0 \tag{4.7}$$

where **V** is a $2nx6$ matrix. We have 6 variables in **b**, so we need atleast 6 equations to solve for b, or in other words, if $n \geq 3$, a unique solution for **b** can be found, defined up to a scale. If $n = 2$, the condition $\gamma = 0$ can be applied, which makes B_{12} =0, on simplifying (3.7). This can be added as an additional equation to system (3.12). The solution to (3.13) is the eigenvector corresponding to the least eigenvalue of $\mathbf{V}^T\mathbf{V}$, or equivalently the right singular vector corresponding to the least singular value of **V**. Once **b** is found, the intrinsic camera matrix **A** can be found.

4.2 Maximum likelihood estimation

The preceding section describes a minimization of an algebraic distance. This can be refined through maximum likelihood estimation. Say we have n images of a model plane, and we have m points on the model plane in each image. We assume that the images are corrupted by independent and identically distributed noise. The following expression needs to be minimized:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} ||\vec{m}_{ij} - \hat{m}(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \vec{w}_j)||^2 \tag{4.8}$$

where $\hat{m}(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \vec{w}_j)$ is the projection of the world point $\vec{w}_j$ on the image plane i according to (2.10). Since there are n images, values of i range from 1 to n. Minimizing this expression is a non-linear minimization problem, solved by the Levenberg-Marquardt algorithm. The initial estimate is taken as the value of **A** found from the method described in the previous section.

Appendices

A Eigenvalues, Eigenvectors and SVD

If **A** is a linear transformation from a vector space V over a field F into itself and **v** is a vector in V that is not the zero vector, then **v** is an eigenvector of **A** if **Av** is a scalar

multiple of **v**. This condition can be written as the equation

$$\mathbf{Av} = \lambda \mathbf{v} \tag{A.1}$$

If **A** is a symmetric n x n matrix, then it can be decomposed as

$$\mathbf{A} = \mathbf{VUV}^T \tag{A.2}$$

where **V** is a n x n matrix of the eigenvectors of **A** arranged column-wise, and **U** is a diagonal n x n matrix with diagonal entries as the eigenvalues of **A**. **V** forms an orthogonal basis of $\mathbf{R}^n$. Thinking of matrices as linear transformations, this can be thought of as a mapping from an orthogonal basis in $\mathbf{R}^n$ to another basis in $\mathbf{R}^n$, with the diagonal entries in in **U** dilating or compressing the elements of the original basis to form the new one.

An SVD can similarly be thought of as a mapping from $\mathbf{R}^n$ to $\mathbf{R}^m$. For an arbitrary m x n matrix **A**, there exist orthogonal matrics **U** and **V** and a diagonal matrix Σ such that

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \tag{A.3}$$

This is called the *singular value decomposition* of **A**. **U** is m x m and **V** is n x n, and form bases for $\mathbf{R}^m$ and $\mathbf{R}^n$ respectively.

B Least squares method

Consider the system

$$\mathbf{Ax} = \mathbf{y} \tag{B.1}$$

where **A** is m x n, **x** is an unknown n x 1 vector and **y** is a known m x 1 vector. If $m = n$ and the rows of **A** are linearly independent, the system is exactly determined and consistent. If $m > n$, the system has more equations than constraints, and is called overdetermined. No consistent solution exists unless there is such redundancy that reduces it to the determined case. The closest solution is found, where the error (**y** - **Ax**) is minimized in the least-squares sense. Mathematically, we find **x** such $||\mathbf{y} - \mathbf{Ax}||^2$ is minimized.

$$\begin{aligned} ||\mathbf{y} - \mathbf{Ax}||^2 &= (\mathbf{y} - \mathbf{Ax})^T(\mathbf{y} - \mathbf{Ax}) \\ &= (\mathbf{y}^T - \mathbf{x}^T\mathbf{A}^T)(\mathbf{y} - \mathbf{Ax}) \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{Ax} - \mathbf{x}^T\mathbf{A}^T\mathbf{y} + \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} \end{aligned} \tag{B.2}$$

Differentiating with respect to **x** gives

$$\frac{\partial ||\mathbf{y} - \mathbf{Ax}||^2}{\partial \mathbf{x}} = 2(-\mathbf{A}^T\mathbf{b} + \mathbf{A}^T\mathbf{Ax}) \tag{B.3}$$

This yields

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{B.4}$$

Again, this solution holds for $\mathbf{b} \neq 0$. If $\mathbf{b} = 0$ then **x** is the solution to $\mathbf{A}^T\mathbf{Ax} = 0$. The closest solution is the eigenvector corresponding to the least eigenvalue of $\mathbf{A}^T\mathbf{A}$.

C Levenberg-Marquardt algorithm

The LMA is an iterative procedure for numeric minimization. Given a set of data points (x_i, y_i), the problem is to find parameters $\boldsymbol{\beta}$ of the model curve $f(x, \boldsymbol{\beta})$ so that the sum of the squares of the deviations is minimized:

$$\hat{\beta} = \text{argmin}_\beta \, S(\boldsymbol{\beta}) \equiv \text{argmin}_\beta \sum_{i=1}^{m} (y_i - f(x_i, \boldsymbol{\beta}))^2 \tag{C.1}$$

In more general terms, we need to minimize a function of the form

$$f(\vec{x}) = \frac{1}{2} \sum_{j=1}^{m} {r_j}^2(\vec{x}) \tag{C.2}$$

where $\vec{x} = (x_1, x_2 \dots x_n)$ is an n dimensional vector, and each $r_j(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ is a function of $\vec{x}$ called a residual. It is assumed that $m \geq n$. We define $\vec{r}\,(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^m$ as

$$\vec{r}\,(x) = (r_1(\vec{x}), r_2(\vec{x}) \dots r_m(\vec{x})) \tag{C.3}$$

Now f can be written as

$$f(\vec{x}) = \frac{1}{2} ||\vec{r}\,(\vec{x})||^2 \tag{C.4}$$

First, let us consider the case when each r_i is linear. Then, f can be written as

$$f(\vec{x}) = \frac{1}{2} ||\mathbf{J}\vec{x} + \vec{r}\,(0)||^2 \tag{C.5}$$

where J is the Jacobian matrix. This case is fully covered in appendix B. In the case when $\vec{r}$ is not linear, we have

$$\nabla f(\vec{x}) = \sum_{j=1}^{m} r_j(\vec{x}) \nabla r_j(\vec{x}) = \mathbf{J}^T \vec{r}\,(x) \tag{C.6}$$

$$\nabla^2 f(\vec{x}) = \mathbf{J}^T \mathbf{J} + \sum_{j=1}^{m} r_j(\vec{x}) \nabla^2 r_j(\vec{x})) \tag{C.7}$$

If it is possible to approximate the residuals as linear (i.e. $\nabla^2 r_j(\vec{x})$ are small), then we can obtain the Hessian ($\nabla^2 f(\vec{x})$) as

$$\nabla^2 f(\vec{x}) = \mathbf{J}^T \mathbf{J} \tag{C.8}$$

With this background, we come back to the problem. To start a minimization, an initial guess $\boldsymbol{\beta}$ is provided, and in each iteration step, the parameter vector $\boldsymbol{\beta}$ is replaced by a new estimate $\boldsymbol{\beta} + \boldsymbol{\delta}$. To determine $\boldsymbol{\delta}$, the functions $f(x_i, \boldsymbol{\beta} + \boldsymbol{\delta})$ are replaced by their linearizations.

$$f(x_i, \boldsymbol{\beta} + \boldsymbol{\delta}) \approx f(x_i, \boldsymbol{\beta}) + J_i \boldsymbol{\delta} \tag{C.9}$$

where

$$J_i = \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \tag{C.10}$$

This reduces $S(\boldsymbol{\beta} + \boldsymbol{\delta})$ to the approximation

$$S(\boldsymbol{\beta} + \boldsymbol{\delta}) \approx \sum_{i=1}^{m} (y_i - f(x_i, \boldsymbol{\beta}) - J_i \boldsymbol{\delta})^2 \tag{C.11}$$

In vector notation,

$$\begin{aligned} S(\boldsymbol{\beta} + \boldsymbol{\delta}) &\approx ||\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}||^2 \\ &= (\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta})^T (\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}) \\ &= ((\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta}))^T - (\mathbf{J}\boldsymbol{\delta})^T)(\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta}) - \mathbf{J}\boldsymbol{\delta}) \\ &= (\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta}))^T (\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta})) - (\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta}))^T \mathbf{J}\boldsymbol{\delta} - (\mathbf{J}\boldsymbol{\delta})^T (\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta})) + (\mathbf{J}\boldsymbol{\delta})^T \mathbf{J}\boldsymbol{\delta} \end{aligned}$$

$$\frac{\partial S(\boldsymbol{\beta} + \boldsymbol{\delta})}{\partial \boldsymbol{\delta}} = -2(\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta}))^T \mathbf{J} + 2\boldsymbol{\delta}^T \mathbf{J}^T \mathbf{J} = 0 \tag{C.12}$$

$$\mathbf{J}^T (\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta})) = \mathbf{J}^T \mathbf{J}\boldsymbol{\delta} \tag{C.13}$$

This is a set of linear equations that can be solved for $\boldsymbol{\delta}$. Levenberg's contribution was to replace this equation with a damped version

$$[\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}]\boldsymbol{\delta} = \mathbf{J}^T (\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta})) \tag{C.14}$$

Ultimately, the goal is to reduce S to its minimum value. λ is a non-negative damping factor that controls the value of $\boldsymbol{\delta}$ to be added at every iteration to $\boldsymbol{\beta}$. So if the reduction in S is rapid, then a smaller value of λ can be used so that we do not overshoot the required minimum value of S, bringing the method closer to the Gauss-Newton method of iterative minimization by bringing the factor $(\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I})\boldsymbol{\delta}$ closer to the Hessian. If the reduction in S is slow, λ can be increased, making the method more similar to the gradient-descent method, because it damps the Hessian.

If the damping factor is large, the Hessian $\mathbf{J}^T \mathbf{J}$ is not used at all as the term $\lambda \mathbf{I}$ is large. This would cause the step to be large where the gradient is large, and small when the gradient is small. Marquardt provided the insight that we can scale each component of the gradient by the curvature, so that there is larger movement along directions where the gradient is smaller. This avoids slow convergence in the direction of small gradient. The identity matrix in (C.7) is replaced by the diagonal of the Hessian.

$$\boldsymbol{\delta} = [\mathbf{J}^T \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{J})]^{-1} \mathbf{J}^T (\mathbf{y} - \mathbf{f}(\mathbf{x}, \boldsymbol{\beta})) \tag{C.15}$$