Back to the Future

by

Marty McFly

B.S. in Computer Science and Engineering
Massachusetts Institute of Technology (2009)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2011

© Massachusetts Institute of Technology 2011. All rights reserved.

Author . . . .
Department of Electrical Engineering and Computer Science
May 20, 2011

Certified by . . . .
Dr. Emmet Brown
Associate Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by . . . .
Dr. Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Back to the Future

by

Marty McFly

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2011, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The text of your abstract goes here. It will be single-spaced and the rest of the text that is supposed to go on the abstract page will be generated by the abstractpage environment.

Thesis Supervisor: Dr. Emmet Brown
Title: Associate Professor of Media Arts and Sciences

Acknowledgments

The text of your acknowledgements goes here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This is the introduction.

Chapter 2

Citations and Images

Section 2.1 presents citations example. Section 2.2 introduces an example of including an image in a Figure context.

2.1 Citations

This section includes various citations examples. For example, one protagonist of the thesis this document is based on is the MDS robot platform [5]. The MDS has many sensors, including a Firefly Firewire camera [6]. Another feature of the MDS platform is its IRCP network communication protocol [3].

The Camshift (Continuously Adaptive Mean Shift) algorithm is described in the article [2]. Finally, citations can also be included in foot notes. [1]

2.2 Images

Figure 2-1 shows a sample image.

[1] OpenCV [1] implements the extended version of the Viola-Jones detector described by Lienhart and Maydt [4].

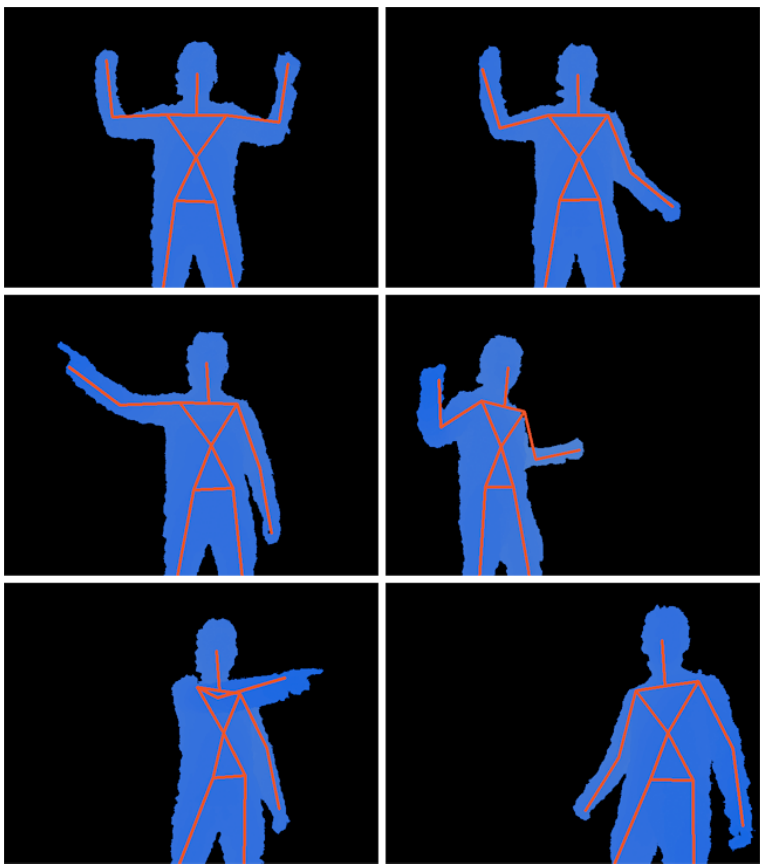

Figure 2-1: Example of including an image inside a `figure` using command `includegraphics`.

Chapter 3

Tables

This chapter presents examples of how to include tables. Table 3.1 shows a one column table while table 3.2 shows a two column table.

Table 3.1: Example of a one column table

Camera
`startCapturing`
`startCalibration`
`grabImage`
`getWidth`
`getHeight`
`getTimestamp`
`getFrameRate`

Table 3.2: Example of a two column table

Color Model

Color Model	Description
`BGR`	Blue-Green-Red channels
`BGRA`	Blue-Green-Red-Alpha channels
`RGB`	Red-Green-Blue channels
`RGBA`	Red-Green-Blue-Alpha channels
`GRAY`	Grayscale channel

Finally, Table 3.3 shows how can a table be used to describe an algorithm.

Table 3.3: Example of using a table to describe an algorithm

```
CALIBRATE (currentImages, requiredImages):
If (currentImages < requiredImages)
    Then:
        Search checkerboard pattern in color image;
        Search checkerboard pattern in amplitude image;
        If the pattern is found in both images
            Then:
                Extract corners from color image and save them;
                Extract corners from amplitude image and save them;
                Increment currentImages counter;
    Else:
        Run color camera calibration with saved corners;
        Run depth camera calibration with saved corners;
        Compute relative transformation with computed parameters;
```

Chapter 4

Math

This chapter shows different ways of including math content. It also show some `subsection` examples.

4.1 Simple math content

This paragraph has math variables inline with the sentences. If f is the focal length of a camera, and m_x, m_y are the ratios of pixel width and pixel height per unit distance, respectively, then (f_x, f_y) represents the focal length expressed in units of horizontal and vertical pixels.

4.1.1 Using `align` context

A math formula can also be included on its own line:

$$(f_x, f_y) = (f m_x, f m_y) \tag{4.1}$$

4.2 More complex math content

This section includes more paragraphs with more examples, including math formulas with labels and references to those labels.

4.2.1 Examples

Furthermore, (x_0, y_0) represents the principal point of the camera. Assuming that the skew coefficient between the x and y axes is zero, the intrinsic matrix of this camera is given by 4.2:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

The calibration process also provides a rotation matrix and a translation vector. These parameters describe the transformation between the world's coordinate system and the camera's coordinate system. If $\mathbf{R}$ denotes the 3×3 rotation matrix, and $\mathbf{t}$ denotes the 3×1 translation vector, the extrinsic matrix of this camera is given by the 3×4 matrix

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \tag{4.3}$$

Therefore, the synchronized calibration algorithm outputs two intrinsic matrices, $\mathbf{K}_d$ and $\mathbf{K}_c$, and two extrinsic matrices, $[\mathbf{R}_d\ \mathbf{t}_d]$ and $[\mathbf{R}_c\ \mathbf{t}_c]$, where the d and c subscripts are used to differentiate between the depth camera's parameters and the color camera's parameters. While the intrinsic matrices contain information about the internals of the camera, the extrinsic matrices hold information about how the cameras are positioned in space. This information is used in the next section to compute the relative transformation between both cameras.

When computing the relative transformation between the cameras, the direction of the transformation is chosen to be from the depth camera to the color camera. As discussed in Section **??**, the field of view of the depth camera is within the field of view of the color camera. Therefore, every point in the depth image will have a corresponding point in the color image, but not necessarily vice versa.

If $\mathbf{P} = (X, Y, Z)^T$ is a point in world coordinates, the position of $\mathbf{P}$ in the depth camera's coordinate system is given by $\mathbf{q}_d$. Similarly, the position of $\mathbf{P}$ in the color camera's coordinate system is given by $\mathbf{q}_c$. The points $\mathbf{q}_d$ and $\mathbf{q}_c$ can be expressed in terms of the cameras' extrinsic parameters by equations (4.4) and (4.5), respectively.

$$\mathbf{q}_d = \mathbf{R}_d \mathbf{P} + \mathbf{t}_d \tag{4.4}$$

$$\mathbf{q}_c = \mathbf{R}_c \mathbf{P} + \mathbf{t}_c \tag{4.5}$$

Considering now the image of $\mathbf{P}$ in the depth image as having coordinates (x_d, y_d), this point can be expressed in homogeneous coordinates as $\mathbf{p}_d = (w\,x_d, w\,y_d, w)^T$, for some constant w. Using the depth camera's intrinsic parameters, $\mathbf{p}_d$ can be expressed by the equation:

$$\mathbf{p}_d = \mathbf{K}_d \mathbf{q}_d \tag{4.6}$$

This automatically reveals another expression for $\mathbf{q}_d$:

$$\mathbf{q}_d = \mathbf{K}_d^{-1} \mathbf{p}_d \tag{4.7}$$

Combining the two expressions for $\mathbf{q}_d$ (equations (4.4) and (4.7)), and solving for $\mathbf{P}$ gives an equation for point $\mathbf{P}$:

$$\begin{aligned}
\mathbf{K}_d^{-1} \mathbf{p}_d &= \mathbf{R}_d \mathbf{P} + \mathbf{t}_d \\
\mathbf{R}_d \mathbf{P} &= \mathbf{K}_d^{-1} \mathbf{p}_d - \mathbf{t}_d \\
\mathbf{P} &= \mathbf{R}_d^{-1} \mathbf{K}_d^{-1} \mathbf{p}_d - \mathbf{R}_d^{-1} \mathbf{t}_d
\end{aligned} \tag{4.8}$$

This expression for $\mathbf{P}$ can be substituted in equation (4.5) to get a new expression for $\mathbf{q}_c$:

$$\begin{aligned}
\mathbf{q}_c &= \mathbf{R}_c (\mathbf{R}_d^{-1} \mathbf{K}_d^{-1} \mathbf{p}_d - \mathbf{R}_d^{-1} \mathbf{t}_d) + \mathbf{t}_c \\
&= \mathbf{R}_c \mathbf{R}_d^{-1} \mathbf{K}_d^{-1} \mathbf{p}_d - \mathbf{R}_c \mathbf{R}_d^{-1} \mathbf{t}_d + \mathbf{t}_c
\end{aligned} \tag{4.9}$$

Using equation (4.7), equation (4.9) simplifies to:

$$\mathbf{q}_c = (\mathbf{R}_c \mathbf{R}_d^{-1})\, \mathbf{q}_d + (\mathbf{t}_c - \mathbf{R}_c \mathbf{R}_d^{-1} \mathbf{t}_d) \tag{4.10}$$

Equation (4.10) reveals how the world points in the depth camera's coordinate system are related to the world points in the color camera's coordinate system. As seen from the equation, this transformation is given in terms of the cameras' extrinsic parameters. Therefore, the relative transformation between the depth and color cameras is defined by the rotation matrix in equation (4.11) and the translation vector in equation (4.12).

$$\mathbf{R}_r = \mathbf{R}_c \mathbf{R}_d^{-1} \tag{4.11}$$

$$\mathbf{t}_r = \mathbf{t}_c - \mathbf{R}_r \mathbf{t}_d \tag{4.12}$$

The fusion algorithm must convert every point (x_d, y_d) in the depth image into a point (x_c, y_c) in the color image. This is achieved by first computing the world point $\mathbf{q}_d$ from the depth image point (x_d, y_d). Then, $\mathbf{q}_d$ is transformed into $\mathbf{q}_c$ using the results from Section **??**. Finally, world point $\mathbf{q}_c$ is converted into a color image point (x_c, y_c).

If (f_x, f_y) and (x_0, y_0) are the focal length and principal point of the depth camera, respectively, the world point $\mathbf{q}_d = (X, Y, Z)^T$ can be related to the image point (x_d, y_d) through the perspective projection equations (4.13) and (4.14).

$$x_d = f_x \frac{X}{Z} + x_0 \tag{4.13}$$

$$y_d = f_y \frac{Y}{Z} + y_0 \tag{4.14}$$

The depth camera provides the z-component of $\mathbf{q}_d$.[2] The x and y components are

[2] The depth camera can actually provide all components as discussed in Section **??**. However, the noise in these measurements is high and recomputing the x and y components delivers better results.

computed by solving equations (4.13) and (4.14) for X and Y, respectively:

$$X = \frac{Z}{f_x}(x_d - x_0) \tag{4.15}$$

$$Y = \frac{Z}{f_y}(y_d - y_0) \tag{4.16}$$

Using the color camera's intrinsic parameters, $\mathbf{p}_c$ can be expressed by the equation

$$\mathbf{p}_c = \mathbf{K}_c \mathbf{q}_c \tag{4.17}$$

Furthermore, equation (4.10) gives an expression for $\mathbf{q}_c$. Therefore, combining (4.17) and (4.10) results in a new equation for $\mathbf{p}_c$:

$$\mathbf{p}_c = \mathbf{K}_c(\mathbf{R}_r \mathbf{q}_d + \mathbf{t}_r) \tag{4.18}$$

By expressing $\mathbf{q}_d$ in homogeneous coordinates (that is, $\mathbf{q}_d^{'} = (X, Y, Z, 1)^T$), equation (4.18) can be rewritten as

$$\mathbf{p}_c = \mathbf{K}_c[\mathbf{R}_r \ \mathbf{t}_r]\mathbf{q}_d^{'} \tag{4.19}$$

The image coordinates (x_c, y_c) are obtained by dividing the first and second components of $\mathbf{p}_c$ by its third component. That is, if $\mathbf{p}_c = (x, y, z)^T$, then

$$(x_c, y_c) = \left(\frac{x}{z}, \frac{y}{z}\right) \tag{4.20}$$

Chapter 5

Conclusion

This is the conclusion.

Appendix A

Appendix with One Figure

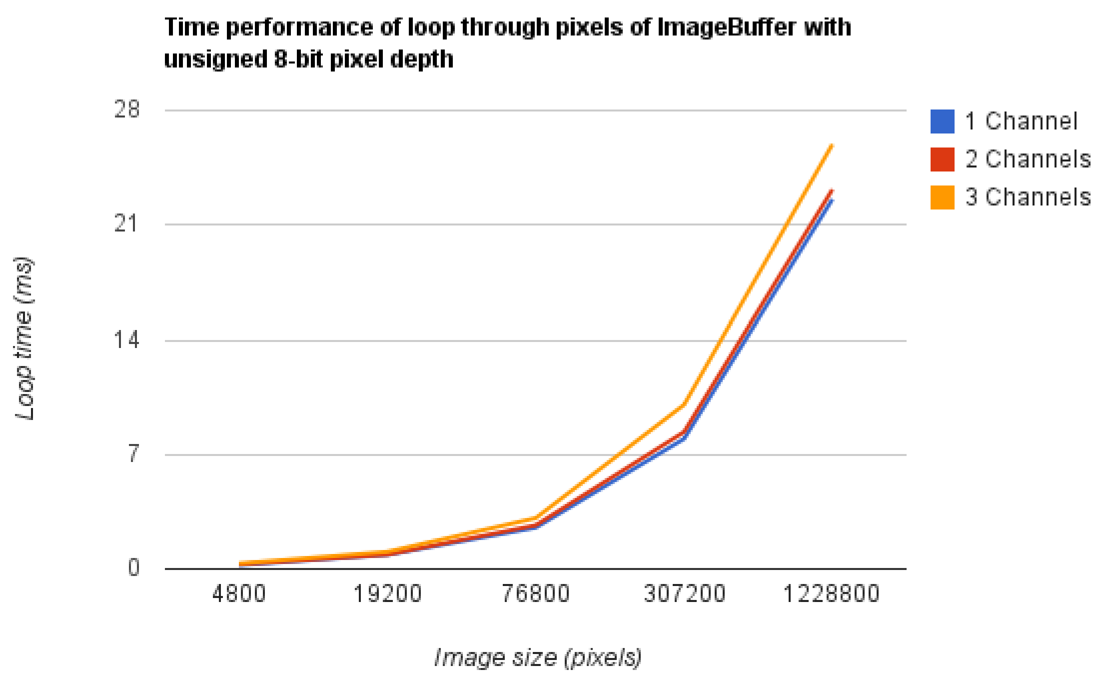

Figure A-1: Time performance of loop through all the pixels of an image buffer. The tests were ran on image buffers of pixel depth BYTE, with one, two, and three channels. The tested image sizes were 80 x 60, 160 x 120, 320 x 240, 640 x 480, and 1280 x 960.

Bibliography

[1] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. Sebastopol, CA: O'Reilly, 2008.

[2] G. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal, Q2*, 1998.

[3] M. Hancher, "A motor control framework for many-axis interactive robots," Master of Engineering thesis, Massachusetts Institute of Technology, Cambridge, MA, May 2003.

[4] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *IEEE ICIP*, 2002, pp. 900–903.

[5] MDS. Personal Robots Group, MIT Media Lab. Cambridge, MA. [Online]. Available: http://robotic.media.mit.edu/portfolio/mobile-dexterous-social-robots

[6] *Firefly MV Product Datasheet*, fireflymv.pdf, Point Grey. [Online]. Available: http://www.ptgrey.com/firefly-mv-ieee-1394b-firewire-cameras