UNIVERSITÀ DI PISA

Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea Specialistica in Informatica

Tesi di Laurea Specialistica in Informatica

Color to gray conversions for stereo matching

Candidato:

Luca Benedetti

Relatori:

Ing. Massimiliano Corsini
Dott. Paolo Cignoni

Controlelatore:

Prof. Francesco Romani

Anno Accademico 2007/08

Dedicated to my family,
you know who you are.

Contents

Chapter 1

Introduction

This thesis belongs to the Computer Graphics and Computer Vision fields, it copes with the image color to grayscale conversion problem with the purpose of improving the quality and the accuracy of the results when the grayscale conversion is applied in the context of stereo matching.

Many different state of the art color to grayscale conversion algorithms have been evaluated, implemented and tested inside the stereo matching context, and a new ad-hoc algorithm that optimizes the conversion process by simultaneously evaluating the whole set of images that has to be matched has been proposed.

1.1 The project

The color to grayscale problem is a dimensionality reduction problem whose importance is often undervalued. For example, as it can be seen in Figure 1.1, isoluminant color changes are not preserved with traditional color to grayscale conversions. In recent years many methods have been proposed, mainly in the context of reproduction of the color image with grayscale mediums. Perceptual accuracy is often the only objective of these results.

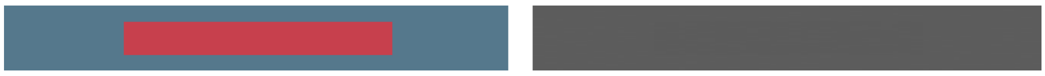


Figure 1.1: *Isoluminant changes are not preserved with traditional color to grayscale conversion. Converting an image of a reddish rectangle whose luminance matches that of the blue background to grayscale results in a featureless image.*

Stereo matching is a field of Computer Vision that tries to resolve the problem of the three dimensional reconstruction of a scene from a pair of two dimensional images. While some standard approaches to this problem use color images, only few standard solutions are able to take real advantage from the color information.

In this thesis we have three objectives:

- To analyze the characteristics of many different state of the art color to gray conversion algorithms.
- To design and implement a new grayscale method based on the existing one that is best suited for our specific needs.
- To evaluate in a thorough way how the choice of different grayscale conversions affects the results computed by the standard stereo matching algorithms.

It is important to underline that the methodological approach follows the *divide et impera* strategy, because it uncouples the color treatment from the stereo matching process using a sepatate preprocessing step for the conversion.

1.2 Thesis structure

After the current introduction Chapter, the thesis is structured in the following way:

- In Chapter 2 the minimal theoretical background will be provided.
- In Chapter 3 the state of the art in color to gray conversions is presented, together with our proposed approach.
- Chapter 4 contains a description of the various implementations.
- In Chapter 5 the experimental evaluation is detailed and the results are presented.
- In Chapter 6 the conclusion are shown and future works are outlined.

1.3 An important note about the images

Even if high quality printers are used, few specialized printing mediums can adequately reproduce the subtly visible changes of colors and grayscale intensities that we show in the images of this work.

Just to make an example, the significant differences between images (b) and (c) in Figure 4.2 on page 46 are almost completely lost when printed on paper, even if they are easily discernible when seen on a standard computer monitor. Thus, we strongly recommend to watch the images on a (preferably calibrated) computer screen.

To ensure the availability of the electronic format, a *Portable Document Format has been proposed* (`pdf`) version of this work is publicly downloadable from the web pages of the *Electronic Theses and Dissertation* service of the University of Pisa at the address: `http://etd.adm.unipi.it` (as of April 2009).

Chapter 2

Background

The background of this work is mainly rooted in the fields of color science, computer vision and image processing. We will now introduce briefly the basics of these topics, referencing to more complete sources at the beginning of each topic section.

In Section 2.1 some basic color theory concepts are introduced, like luminance (2.1.1), color models and color spaces (2.1.2, 2.1.3), gamma (2.1.4) and grayscale conversions (2.1.5). We also enumerate, with a brief explanation, many of the color spaces used by the grayscale conversions that we examined and implemented (see Chapter 3 and Chapter 4).

In Section 2.2 we give a brief overview about the image based 3D reconstruction world: after a quick description of the problem we introduce the stereo matching family of algorithms (2.2.1), then we hint about the multi view stereo matching problem (2.2.2) and eventually we cope with the grayscale conversion problem in the context of stereo matching (2.2.3).

2.1 Luminance and chrominance

Visible light is the electromagnetic radiation of a wavelength that is visible to the human eye in the frequency range between 380 and 750 *nm*. The primary properties of light are: *intensity*, *frequency*, *polarization* and the *wave-particle duality*. Here we will consider just the distribution of light energy versus wavelength, that is the *spectrum*.

A really good introduction to this topic can be found in [1].

2.1.1 Luminance

Luminance is a photometric measure (in $\frac{cd}{m^2}$) of the luminous intensity per unit area of light traveling in a given direction. It describes the amount of light that passes through or is emitted from a particular area, and falls within a given solid angle. The luminance indicates how much luminous power will be perceived by an eye looking at the surface from a particular angle of view. Luminance is thus an indicator of how bright a surface will appear.

2.1.2 Color

Color is the visual perceptual property corresponding in humans to the categories called *red*, *yellow*, *blue* and others. Color derives from the spectrum, interacting in the eye with the spectral sensitivities of the light receptors. Color categories and physical specifications of color are also associated with objects, materials, light sources, etc., based on their physical properties such as light absorption, reflection, or emission spectra.

As already said, the visible spectrum is the portion of the electromagnetic spectrum that is visible to the human eye. A typical human eye will respond to wavelengths from about 380 to 750 nm. In terms of frequency, this corresponds to a band in the vicinity of 790–400 tera hertz.

The spectrum distribution defines the color perceived by the eye, but it does not contain all the colors that the human eyes and brain can distinguish. Unsaturated colors such as pink, and purple colors such as magenta are absent, for example, because they can only be made by a mix of multiple wavelengths. Another, often problematic, characteristic of the color is that different spectrum distributions can be perceived as the same color, this phenomenon is called *metamerism*.

The color of an object depends upon its temperature as predicted by Planck's formula. This law describes the "ideal" blackbody radiation given off by any *blackbody* object above absolute zero. In physics, a black body is an object that absorbs all electromagnetic radiation that falls on it. Nonblack objects emit according to the "ideal" blackbody radiation spectrum multiplied (frequency by frequency) by the proper absorption coefficient characteristic of its surface.

2.1.3 Color Models and Color Spaces

There are several ways to formally describe colors. A *color model* is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four color components. When this model is associated with a precise description of how the components are to be interpreted, the resulting set of colors is called *color space*.

Adding a certain mapping function between the color model and a certain reference color space results in a definite "footprint" within the reference color space. This is known as a *gamut*, and, in combination with the color model, defines a new color space.

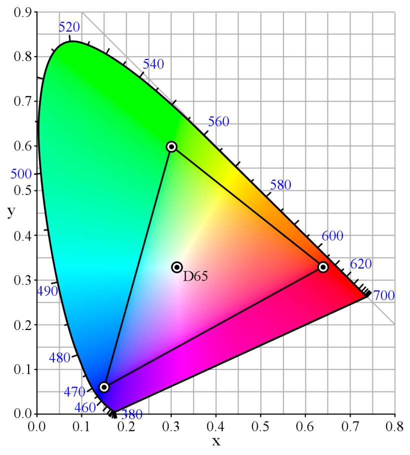


Figure 2.1: *CIE 1931 xy chromaticity diagram showing the gamut of the sRGB color space and location of the primaries. The D65 white point is shown in the center.*

2.1.3.a Tristimulus models

The human eye has receptors, called *cone cells*, for short, middle and long wavelengths. Thus in principle, three parameters can describe a color sensation. The tristimulus values of a color are the amounts of three primary colors in a three-component additive color model.

Any specific method for associating tristimulus values with each color is called a color space. CIE XYZ, one of many such spaces, is special because it is based on direct measurements of human visual perception, and serves as the basis from which many other color spaces are defined.

Absolute color spaces needs to define a *white point*. A white point (often referred to as “reference white” or “target white” in technical documents) is a set of tristimulus values or chromaticity coordinates that serve to define the color “white”. Depending on the application, different definitions of white are needed to give acceptable results. For example, photographs taken indoors may be lit by incandescent lights, which are relatively orange compared to daylight. Defining “white” as daylight will give unacceptable results when attempting to color correct a photograph taken with incandescent lighting.

2.1.3.b CIE XYZ color space

The *CIE 1931 XYZ color space*[1] is one of the first mathematically defined color spaces. It was derived from a series of experiments done in the late 1920s by W. David Wright [2] and John Guild [3]. Their experimental results were combined into the specification of the *CIE RGB* color space[2], a previous attempt from which the CIE XYZ color space was derived.

In the CIE XYZ color space, the tristimulus values are not the responses of the human eye, but rather a set of tristimulus values called X, Y, and Z, which are a sort of “derived” parameters from the red, green and blue colors. This derivation has also been choosed in a way that the Y value can represent the achromatic luminance.

This color space is used directly in the CIE Y grayscale conversion (see Section 3.1.2.a and Section 4.3.1) and as a base for many other methods.

2.1.3.c RGB color model

The *RGB color model* is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors.

The main purpose of the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography. Before

[1]Also known as CIE 1931 color space, CIE stands for *Commission internationale de l'éclairage*, the original French name of the *International Commission on Illumination.*

[2]Not to be confused with the RGB color model.

the electronic age, the RGB color model already had a solid theory behind it, based in human perception of colors.

Unlike the CIE color spaces, that strives to be independent from the reproduction devices, the RGB color model is a device-dependent color space: different devices detect or reproduce a given RGB value differently, since the color elements (such as phosphors or dyes) and their response to the individual R, G, and B levels vary from manufacturer to manufacturer, or even in the same device over time. Thus an RGB value does not define the same color across devices without some kind of color management.

This color space is used directly in the RGB Channel Filter grayscale conversion (see Section 3.1.1.b) and as a base for many other methods.

2.1.3.d HSV and HSL representations

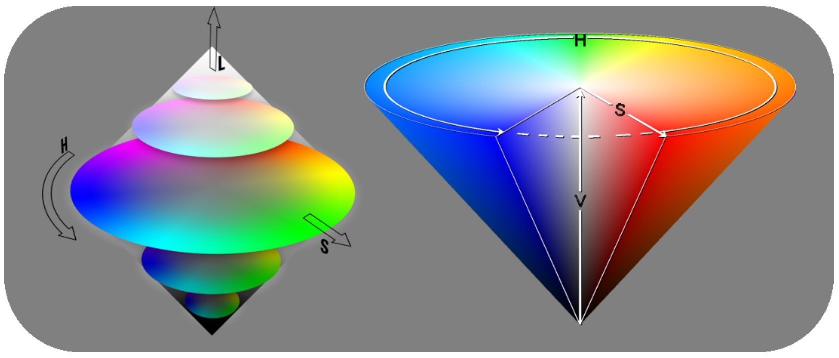


Figure 2.2: *HSL arranged as a double-cone and the conical representation of the HSV model.*

HSL and *HSV* are two related representations of points in an RGB color space, which attempt to describe perceptual color relationships more accurately than RGB, while remaining computationally simple. HSL stands for hue, saturation, lightness, while HSV stands for hue, saturation, value. Both

HSL and HSV describe colors as points in a cylinder whose central axis ranges from black at the bottom to white at the top with neutral colors between them, where angle around the axis corresponds to “hue”, distance from the axis corresponds to “saturation”, and distance along the axis corresponds to “lightness”, “value”, or “brightness”.

The two representations are similar in purpose, but differ somewhat in approach. Both are mathematically cylindrical, but while HSV can be thought of conceptually as an inverted cone of colors (with a black point at the bottom, and fully-saturated colors around a circle at the top), HSL conceptually represents a double-cone or sphere (with white at the top, black at the bottom, and the fully-saturated colors around the edge of a horizontal cross-section with middle gray at its center). Note that while “hue” in HSL and HSV refers to the same attribute, their definitions of “saturation” differ dramatically.

The HSV color space is used directly in the Value HSV grayscale conversion (see Section 3.1.1.a), while the HSL color space is used directly in the Lightness HSL grayscale conversion (see Section 3.1.1.c).

2.1.3.e sRGB color space

sRGB is a standard RGB color space created cooperatively by HP and Microsoft for use on monitors, printers, and the Internet. It uses the *ITU-R BT.709-5* primaries, the same used in studio monitors and HDTV, and a transfer function (gamma curve) typical of CRTs. his specification allows sRGB to be directly displayed on typical monitors, a factor which greatly aided its acceptance.

The sRGB color space has another peculiarity: its gamma[3] can not be

[3]See Section 2.1.4 for the definition of the gamma in general and Section 2.1.4.a for the specific sRGB gamma.

expressed as a single numerical value.

The sRGB color space has been endorsed by the W3C, Exif, Intel, Pantone, Corel, and many other industry players; it is also used in proprietary and open graphics file formats, such as SVG. As the recommended color space for the Internet, sRGB should be used for editing and saving all images intended for publication to the WWW. The sRGB color space is well specified and is designed to match typical home and office viewing conditions, rather than the darker environment typically used for commercial color matching.

Due to the standardization of sRGB on the Internet, on computers, and on printers, many low- to medium-end consumer digital cameras and scanners use sRGB as the default (or only available) working color space. Used in conjunction with an inkjet printer, an sRGB image produces what is often regarded as satisfactory for home use. However, consumer-level camera LCDs are typically uncalibrated, meaning that even though the image is being labeled as sRGB, one cannot conclude that the image is color-accurate on the LCD.

The two dominant programming interfaces for 3D graphics, OpenGL and Direct3D, have both incorporated sRGB.

Unless otherwise stated, we assume that the input images are encoded in this color space.

2.1.3.f Color opponent process

The color opponent process is a color theory that states that the human visual system interprets information about color by processing signals from cones and rods in an antagonistic manner. The three types of cones have some overlap in the wavelengths of light to which they respond, so it is more efficient for the visual system to record differences between the responses of cones, rather than each type of cone's individual response. The opponent color theory suggests that there are three opponent channels: red versus green, blue versus yellow, and black versus white. The latter type is achromatic and detects light/dark variation, or luminance. -

2.1.3.g CIE L* a* b* color space

The CIE 1976 (L*, a*, b*) color space is a color-opponent space with dimension L for lightness and a and b for the color opponent dimensions, based on nonlinearly compressed CIE XYZ color space coordinates. CIE LAB uses a red/green axis for the a value and a blue/yellow axis for the b value. The intention is to create a space which can be computed via simple formulas from the XYZ space, but is more perceptually uniform than XYZ. Perceptually uniform means that a change of the same amount in a color value should produce a change of about the same visual importance. When storing colors in limited precision values, this can improve the reproduction of tones.

CIE LAB is relative to the white point of the XYZ data it were converted from. Lab values do not define absolute colors unless the white point is also specified. Often, in practice, the white point is assumed to follow a standard and is not explicitly stated, for "absolute colorimetric" rendering intent the values are relative to CIE standard illuminant D50[4].

The lightness correlate in CIE LAB is calculated using the cube root of the relative luminance.

The CIE LAB color space is used, among others, in the Fairchild Lightness grayscale conversion (see Section 3.1.3.a) and in the Gooch Color2Gray grayscale conversion (see Section 3.2.2).

[4]All these topics are best explained in [1].

2.1.3.h CIE L* u* v* color space

The CIE 1976 (L*, u*, v*) color space, is another simple-to-compute transformation of the 1931 CIE XYZ color space, but which attempted perceptual uniformity. CIE LUV uses chromacity (saturation) for the u value and hue angle for the v value. It is extensively used for applications such as computer graphics which deal with colored lights.

The CIE LUV color space is used in the Nayatani Lightness VAC, in the Nayatani Lightness VCC and in the Smith Apparent grayscale conversions (see Section 3.1.3.b, Section 3.1.3.c and Section 3.2.8).

2.1.4 Gamma

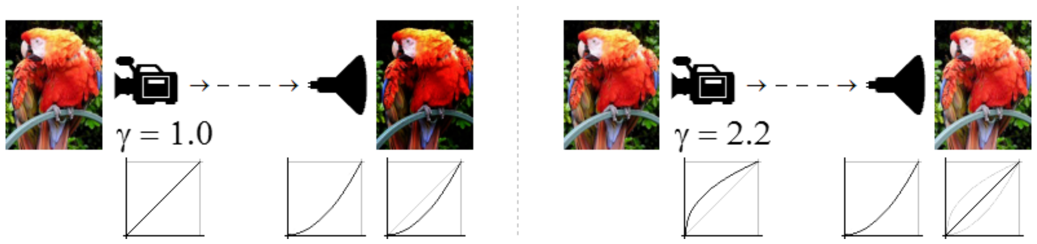


Figure 2.3: *Systems with linear and gamma-corrected cameras. The dashes in the middle represent the storage and transmission of image signals or data files. The three curves represent input–output functions of the camera, the display, and the overall system, respectively.*

Gamma correction is a nonlinear operation used to code and decode luminance or tristimulus values in video or still image systems. Gamma correction is, in the simplest cases, defined by the following power-law expression:

$$V_{\text{out}} = V_{\text{in}}^{\gamma} \tag{2.1}$$

where the input and output values are non-negative real values, typically in a predetermined range such as 0 to 1. A gamma value $\gamma < 1$ is sometimes called an encoding gamma, and the process of encoding with this compressive power-law nonlinearity is called gamma compression; conversely a gamma value $\gamma > 1$ is called a decoding gamma and the application of the expansive power-law nonlinearity is called gamma expansion

Almost all grayscale conversion algorithms must work with linearized images, some conversions are particularly sensitive to this issue (see Section 3.4).

2.1.4.a sRGB gamma

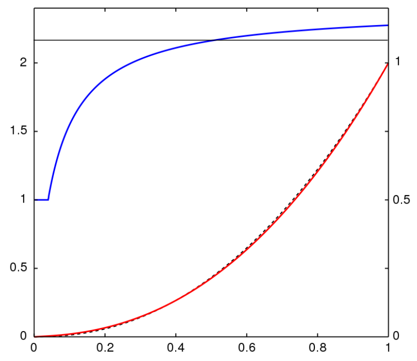


Figure 2.4: *Plot of the sRGB standard gamma-expansion nonlinearity (red), and its local gamma value, slope in log–log space (blue).*

Unlike most other color spaces, the sRGB gamma can not be expressed as a single numerical value. The overall gamma is approximately 2.2, consisting of a linear (gamma 1.0) section near black, and a non-linear section elsewhere involving a 2.4 exponent and a gamma (slope of log output versus log input) changing from 1.0 through about 2.3.

Unless otherwise stated, we use these formulas for gamma encoding and decoding.

2.1.5 Color to gray conversion

Colors in an image may be converted to a shade of gray by calculating the effective brightness or luminance of the color and using this value to create a shade of gray that matches the desired brightness. This may be useful for aesthetic purposes, for printing without colors and for image computations that need (or can be speeded up using) a single intensity value for every pixel.

Color to grayscale conversions perform a reduction of the three dimensional color data into a single dimension. A standard linear technique for dimensionality reduction is *PCA* (principal component analysis). As well explained in [4], PCA is not a good technique for color to gray conversion:

> *In PCA, a set of orthogonal vectors, the principal components, lying along the directions of maximal data variation is found. To reduce the dimensions of the data, a subset of the principal components are used to form a subspace onto which the original data is projected. While preserving the color variation is necessary for pleasing grayscale images, it is not sufficient. If we have an image with most of the colors clustered in two regions, red and green for example, we can envision a "dumbbell"-shaped distribution of pixels in color space. By preserving variance, one end of the "dumbbell" will be mapped to the light end of the grayscale gamut while the other will be mapped to the dark end. If we then examine the histogram of the resulting grayscale image, we will find a large number of empty gray bins in the center of the range. As such, any detail within the red and green "dumbbell "clusters will be reproduced over a very small gray range and may not be perceived.*

It is evident that some loss of information during the conversion is inevitable, so the goal is to save as much information from the original color image as possible.

At the same time, the aim is also to produce “the best” grayscale results. Depending on the context, the best results are either the most perceptually accurate, or the ones that keeps the maximum image information, often at the cost of perceptual accuracy, or the ones that maintain some specific global properties like luminance, dynamic range, contrast, etc.

Recently, various approaches to the color to grayscale conversion problem have been proposed in [5, 6, 7, 8, 4, 9, 10, 11].

2.2 Image based 3D reconstruction

The projection of light rays onto the retina presents our visual system with an image of the world that is inherently two-dimensional, yet we are able to interact with the three-dimensional world, even in situations new to us, or with objects unknown to us. That we accomplish this task easily implies that one of the functions of the human visual system is to reconstruct a 3D representation of the world from its 2D projection onto our eyes.

In computer vision the problem of automatic reconstruction of three-dimensional objects and environments from sets of two or more photographic images is being widely studied, because it is important for many applications that integrate virtual and real data. Many approaches have been used to solve this problem. Traditional methods are based on matching features from sets of two or more images.

In this work we test the impact of different advanced grayscale conversions in the *dense stereo matching* case, so in this section we will introduce this case more in depth with respect to the others.

In traditional stereo methods, for a complete 3D model of real object, many parts of the surface must be computed with respect to a set of base viewpoints, and these surface patches must be fused into a single consistent model. While the classical vision problem of two camera stereo has been studied for decades, in the last few years there has been a surge of interest in shape reconstruction from multiple views. A good reference for both stereo and multi view methods is [12].

In order to compute a 3D reconstruction, a stereo system must first solve the correspondence problem. Matching algorithms can produce sparse or dense disparity maps, depending on exactly what features are used as a basis for the correspondence. In dense stereo matching, the disparity is estimated for every pixel; In contrast, for sparse stereo matching, the disparity is computed at a subset of "interesting" points in the image, called *local features*; The search for correspondence is restricted to a sparse set of detected features.

A local feature is an image pattern which differs from its immediate neighborhood. It is usually associated with a change of an image property or several properties simultaneously, although it is not necessarily localized exactly on this change. The image properties commonly considered are intensity, color, and texture. Local features can be points, but also edges or small image patches. Typically, some measurements are taken from a region centered on a local feature and converted into descriptors. The descriptors can then be used for various applications, including matching. A comprehensive survey on this topic can be found in [13].

2.2.1 Stereo matching

Stereo matching is a problem that has been studied over several decades in computer vision and many researchers have worked at solving it. The proposed approaches can be broadly classified into feature and correlation based approaches.

The simplest case occurs when the images have been rectified in a fronto-

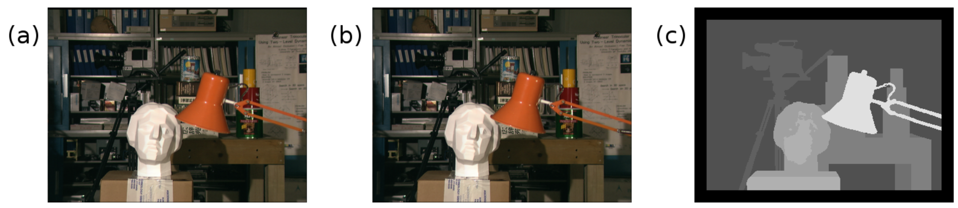


Figure 2.5: *A pair of rectified images (a), (b) and the corresponding disparity map (c).*

parallel position with respect to the object. In this case a dense matching algorithm can compute a map of the horizontal disparity $\mathrm{d}(x, y)$ between the images that is inversely proportional to the distance of every pixel from the camera. The correspondence between a pixel (x, y) in reference image r and a pixel (x', y') in matching image m is then given by

$$x' = x + s\mathrm{d}(x, y), \quad y' = y \tag{2.2}$$

where $s = \pm 1$ is a sign chosen so that disparities are always positive.

A first problem that occurs is how to obtain the rectified image pair. Camera calibration and pixel rectification are necessary, and it really helps to have “well positioned” photos. Then, algorithms performs a series of computation steps that can be roughly aggregated in:

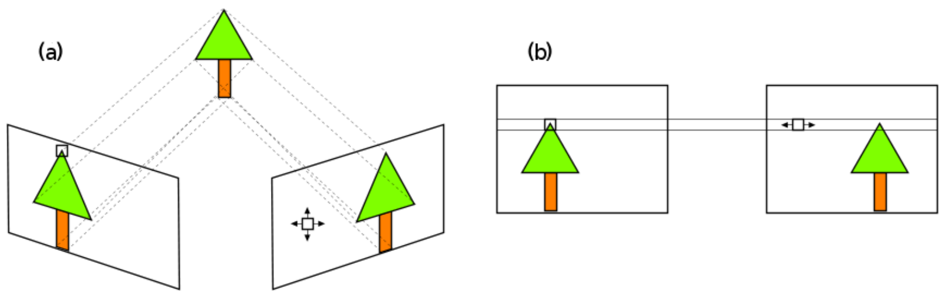


Figure 2.6: *The search space before (a) and after (b) rectification.*

1. matching cost computation
2. cost (support) aggregation
3. disparity computation / optimization
4. disparity refinement

In an effort to measure this subfield's progresses, a general framework with many combinable implementations of these steps has been released by [14]. Many dataset with ground truths have been subsequently produced by the same authors over the subsequent years. We used this framework and these datasets to test the effects of our the various grayscale conversions in the stereo matching results.

2.2.2 Multi view stereo matching

The goal of multi view stereo matching is to reconstruct a complete 3D object model from a collection of images taken from known camera viewpoints. Over the last few years, a number of high quality algorithms have been developed, and the state of the art is improving rapidly.

Multi view stereo matching is beyond the scope of this work, but we are planning to extend the study of grayscale conversions to this case and we are already developing a flexible multi view 3D reconstruction framework, called *PipeLAB*, which will expand upon the ideas of [15] and [16]. PipeLAB can already use the various grayscale conversion implementations as modules (see Section 4.1).

2.2.3 Color and grayscale in matching

Few articles deal with color correlation based matching, the most common approaches are to take the mean of the three color components or to aggregate in some empirical way the information obtained from the single channels.

A grayscale conversion preprocessing step can speed up many image matching algorithms and, in some cases, can give *better* results with respect to color processing. A good grayscale preprocessing can, for example, improve the disparity maps obtained by dense stereo matching algorithms, generate more robust local features in sparse matching algorithms and improve some steps in multi view algorithms[5].

Among the few studies on the correlations between color and grayscale in matching algorithms we can cite [17] and [18]. We are trying to improve upon these studies considering advanced grayscale conversion techniques previously used only for reproduction and aesthetic purposes.

[5]The fusion step, in particular. Since we will not explain these topics here, please refer to [15].

Chapter 3

Grayscale conversions

We have examined and implemented many different existing color to gray conversion algorithms in order to find the best starting point for our specific needs.

In this chapter we will explain many grayscale conversion techniques, starting from ten “simple” ones (Section 3.1) and ending with eight advanced techniques that constitute the state of the art in this field (Section 3.2). Then, we present, in Section 3.3, a new grayscale algorithm that is a variant of the Grundland et al. [7, 8] algorithms (see Section 3.2.4) and is designed specifically for stereo and multi view stereo matching. We eventually discuss the effects of gamma correction in the grayscale conversions in Section 3.4.

3.1 Basic Techniques

Ten “simple” grayscale conversion techniques are explained in this section. All these methods are “simple” in the sense that they are *local* functions of every pixel, e.g. for every pixel of the color image a grayscale value is computed using a function whose only parameters are the values of the corresponding color pixel.

The notation used to specify the function applied to every image pixel is:

$$Y_{xy} = \mathrm{f}(|C_{xy}|) \tag{3.1}$$

where x and y are the pixel's coordinates, Y_{xy} is the grayscale value of that pixel and $|C_{xy}|$ is the vector with the (gamma linearized) color values of that pixel in the starting color space.

The first four (Section 3.1.1) are "quick and dirty" methods, (too) commonly used by existing systems. The next three (Section 3.1.2) are standard methods, good enough for non-specialized uses. The last three (Section 3.1.3) are listed in this section because they are local but they are actually based on more advanced color space topics and can mitigate the problem of isoluminant colors.

3.1.1 Trivial methods

We begin with the most basic methods, where the grayscale value is the result of a trivial function of the color values of the same pixel. These methods loses a lot of the image information because for every pixel they discard one or two of the three color values and blindly average the remaining ones. These color to grayscale conversions should never be used in real life applications, although many systems use them.

We implemented all these methods as PipeLAB modules[1] in order to observe, in future works, the impact of a poorly chosen grayscale conversion in multi-view stereo matching results.

3.1.1.a Value HSV

The worst method that can be considered is: take the HSV representation of the image (see Section 2.1.3.d) and use the "Value" value as the grayscale

[1]See Section 2.2.2 and Section 4.1.

value.

This is equivalent to:

$$Y_{xy} = \max(R_{xy}, G_{xy}, B_{xy}) \tag{3.2}$$

As can be easily seen, this method discards more than two thirds of the original information because for every pixel it discards two of the color values and, in addition, it loses the information relative to which color value is keeped for a pixel. Another evident problem of this approach is that the resulting image luminance is heavily biased toward white.

3.1.1.b RGB Channel Filter

Another quick way to obtain a grayscale image is: choose a channel between R, G or B and use this channel as Y.

This method discards approximately two thirds of the image information, with the green filter giving the best results and the blue filter giving the worst results. This is lightly better than the Value HSV method because the color value choosed is consistent for all the pixels of the image.

In classic photography, this same effect can be achieved using a black and white film and a red, green or blue optic filter.

3.1.1.c Lightness HSL

Another "bad" method is: take the HSL representation of the image (see Section 2.1.3.d) and use the Lightness value as the grayscale value.

This is equivalent to:

$$Y_{xy} = \frac{\max(R_{xy}, G_{xy}, B_{xy}) + \min(R_{xy}, G_{xy}, B_{xy})}{2} \tag{3.3}$$

This method discards more than one third of the original information, because a color value is discarded from every pixel, the remaining values are averaged and it loses the information relative to which color value is discarded for a pixel.

3.1.1.d Naive Mean

The last of these methods, and the best (from the information) between these, is: take the mean between the color channels.

This is equivalent to:

$$Y_{xy} = \frac{R_{xy} + G_{xy} + B_{xy}}{3} \tag{3.4}$$

The advantage of this method with respect to the other trivial ones is that it takes information from every channel, although it does not consider the relative spectral power distribution of the RGB channels.

3.1.2 Direct methods

An easy improvement over the trivial methods is to calculate the grayscale value using a weighted sum over the color channels. Using different weights for different colors, we can take into account factors such as the relative spectral distribution of the color channels and the human perceptive sensibilities. The vast majority of the existing grayscale conversion implementations use a method of this family, often wrongly using the gamma precompressed values instead of the linear ones.

We implemented, as PipeLAB modules[2], two of the most widely used methods: CIE Y and the NTSC Rec.601. The former method has also been choosed as the basic techniques representative for the tests with the stereo matching framework (see Section 4.4). The QT builtin is included here only as an example of grayscale conversion with integer math.

[2]See Section 2.2.2 and Section 4.1.

3.1.2.a CIE Y

A widely used conversion is based on the CIE 1931 color space [2, 3]: take the XYZ representation of the image (see Section 2.1.3.b) and use the Y value as the grayscale value. Assuming that the image is in linearized sRGB (see Section 2.1.4.a) this is equivalent to the following weighted sum over the color values:

$$Y_{xy} = 0.212671R_{xy} + 0.71516G_{xy} + 0.072169B_{xy} \tag{3.5}$$

3.1.2.b NTSC Rec.601

Another widely used conversion is the NTSC Rec.601, created in 1982 by the ITU-R organization for *luma* definition in gamma precompensated television signals:

$$Y_{xy} = 0.299R_{xy} + 0.587G_{xy} + 0.114B_{xy} \tag{3.6}$$

Many existing implementations use these values, both with linear and gamma precompensated images.

3.1.2.c QT builtin

The QT library[3], in the *qGray* function, shows a typical approximation of the NTSC Rec.601 values:

$$Y_{xy} = \frac{11R_{xy} + 16G_{xy} + 5B_{xy}}{32} \tag{3.7}$$

This conversion, equivalent to $Y_{xy} = 0.34375R_{xy} + 0.5G_{xy} + 0.15625B_{xy}$, is designed to work well with integer representation in the $[0 \mathinner{..} 255]$ range, because the division by 32 can be implemented using simple bit shifts. Their implementation ignores any issues regarding gamma compression.

[3]Used as the main support library through all this project development, see [19].

3.1.3 Chrominance direct methods

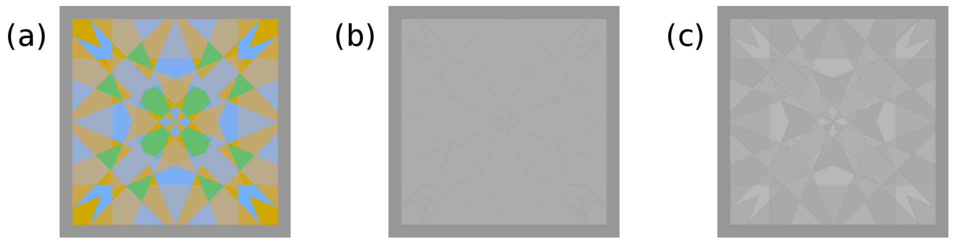


Figure 3.1: *An isoluminant color image (a) converted with the CIE Y method (b) and with the Fairchild Lightness method (c).*

A problem with the previous approaches is that the distinction between two different colors of similar luminance is lost. In this section we consider three methods that are based on more advanced color space topics with respect to the previous ones and try to mitigate this problem. These conversions are still local functions of the image pixels, but they assign different grayscale values to *isoluminant* colors.

To achieve this result the luminance information is slightly altered using the chrominance information. To map different colors to increases or decreases of the "traditional" luminance, a result from human color perception studies is used: the *Helmholtz-Kohlrausch effect* [20, 21, 11].

The Helmholtz-Kohlrausch (H-K) effect states that the perceived lightness of a stimulus changes as a function of the chroma: this phenomenon is predicted by a chromatic lightness term that corrects the luminance based on the color's chromatic component and on starting colorspace.

We examined three such predictors: Fairchild Lightness, Nayatani Lightness VAC and Nayatani Lightness VCC, and we implemented, as a PipeLAB module[4], only the Nayatani Lightness VAC method (3.1.3.b) that is used as

[4]See Section 2.2.2 and Section 4.1.

a base for an advanced technique from Smith et al. [11] explained in Section 3.2.8 and implemented as described in Section 4.3.3.

3.1.3.a Fairchild Lightness

Fairchild's CIE LAB (see Section 2.1.3.g) chromatic lightness metric is fit to Wyszecki 1967 [22] data and is defined as:

$$Y_{xy} = L_{xy} + (2.5 - 0.025L_{xy}) \left(0.116 \left| \sin\left(\frac{H_{xy} - 90}{2}\right) \right| + 0.085\right) C_{xy} \quad (3.8)$$

where L_{xy}, C_{xy} and H_{xy} are the values of a cylindrical representation of the CIE LAB color space called CIE LAB LCH (lightness, chroma, hue angle). For more information see [20].

3.1.3.b Nayatani Lightness VAC

Nayatani [23, 24] defines a chromatic lightness metric based on the CIE LUV color space (see Section 2.1.3.h) and the *Variable-Achromatic-Color* (VAC) approach, in which an achromatic sample's luminance is adjusted to match a color stimulus. VAC was used in the seminal 1954 Sanders-Wyszecki study, and again in Wyszecki's later 1964 and 1967 studies [22].

Chromatic object lightness is predicted by the following equation:

$$Y_{xy} = L_{xy} + \left[-0.1340\,\mathrm{q}\left(\theta_{xy}\right) + 0.0872K_{Br_{xy}}\right] s_{uv_{xy}} L_{xy} \quad (3.9)$$

where $s_{uv_{xy}}$ is a function of u and v that gives the chromatic saturation related to the strength the H-K effect according to colorfulness, the quadrant metric $\mathrm{q}\left(\theta_{xy}\right)$ predicts the change of the H-K effect for varying hues and constant $K_{Br_{xy}}$ expresses the H-K effect's dependence on the eyes' adapting luminance. See [23, 24, 11] for more detailed explanations.

3.1.3.c Nayatani Lightness VCC

Another chromatic lightness metric defined [24] by Nayatani is based on the CIE LUV color space (see Section 2.1.3.h) and the *Variable-Chromatic-Color* (VCC) approach, in which the chromatic content of a color stimulus is adjusted until its brightness matches a given gray stimulus. VCC is less common than VAC.

The chromatic object lightness equation is almost identical to the VAC case:

$$Y_{xy} = L_{xy} + \left[-0.8660\, \mathrm{q}\left(\theta_{xy}\right) + 0.0872 K_{Br_{xy}}\right] s_{uv_{xy}} L_{xy} \tag{3.10}$$

A quantitative difference between them is that VCC lightness is twice as strong VAC lightness (in log space). Moreover, in VCC lightness has been observed [24, 11] that its stronger effect maps many bright colors to white, making it impossible to distinguish between very bright isoluminant colors.

3.2 Advanced Techniques

We present, roughly in chronological order, eight advanced techniques that constitute the state of the art in this field.

In this section we summarize briefly the fundamental ideas behind these methods, in Chapter 3 we will explain in more detail the algorithms that we implemented and we will include the relative examples.

For the sake of simplicity we call these method using the surname of one of the authors and a mnemonic adjective taken from the title of the relative paper:

- *Bala Spatial*, in Section 3.2.1, adds high frequency chromatic information to the luminance.

- *Gooch Color2Gray*, in Section 3.2.2, find gray values that best match the original color differences through an objective function minimization process.
- *Rasche Monochromats*, in Section 3.2.3, minimizes an energy function over the colors.
- *Grundland Decolorize*, in Section 3.2.4, find a global continuous mapping that adds lost chromatic information to the luminance channel.
- *Queiroz Invertible*, in Section 3.2.5, transform colors into high frequency textures that are applied onto the gray image and can be later decoded back to color.
- *Alsam Sharpening*, in Section 3.2.6, aims to create sharp grayscale from the original color rather than enhancing the separation between colors.
- *Neumann Adaptive*, in Section 3.2.7, stresses perceptual loyalty by measuring the image's gradient field by color differences in their Coloroid color space.
- *Smith Apparent*, in Section 3.2.8, combines global and local conversions in a way similar to the Alsam Sharpening method.

3.2.1 Bala Spatial

In their short paper studying chromatic contrast for grayscale conversion, Bala et al. [5] take a spatial approach and introduce color contrasts in CIE LAB LCH (already introduced in Section 3.1.3.a) by adding the high-pass filtered chroma channel to the lightness channel. To prevent overshooting in already bright areas, this correction signal is locally adjusted and its sign is taken from the lightness contrast. The algorithm is susceptible to problems in chroma and lightness misalignment.

3.2.2 Gooch Color2Gray

A different approach was taken by Gooch et al. [6], who introduced a local algorithm known as *Color2Gray*. In this gradient-domain method, the gray value of each pixel is iteratively adjusted to minimize an objective function, which is based on local contrasts between all the pixel pairs.

Original contrast between each pixel and its neighbors is measured by a signed distance, whose magnitude accounts for luminance and chroma difference and whose sign represents the hue shift with respect to a user defined hue angle.

The computational complexity of this method is really high: $O(N^4)$, this can be improved by limiting the number of considered differences (e.g. by color quantization). A recent extension to a multi resolution framework by Mantiuk et al. [25] improves the algorithm's performance: in their approach the close neighborhood of a pixel is considered on fine levels of a pyramid, whereas the far neighborhood is covered on coarser levels. The authors claim that this enables them to convert bigger images and perform computations faster.

3.2.3 Rasche Monochromats

Another conversion was introduced by Rasche et al. [4]. Their method aims to preserve contrast while maintaining consistent luminance. The authors formulate an error function based on matching the gray differences to the corresponding color differences. The goal is minimizing the error function to find an optimal conversion. The authors propose using color quantization to reduce the considerable computational costs of the error minimization procedure.

3.2.4 Grundland Decolorize

Grundland and Dodgson [7, 8] proposed the Decolorize algorithm for contrast enhancement as well as converting color to grayscale. They perform a global grayscale conversion by expressing grayscale as a continuous, image dependent, piecewise linear mapping of the primary RGB colors and their saturation.

In their YPQ color space[5], the color differences are projected onto the two *predominant* chromatic contrast axes and are then added to the luminance image. Unlike principal component analysis[6] which optimizes the variability of observations, predominant component analysis optimizes the differences between observations. The predominant chromatic axis aims to capture, with a single chromatic coordinate, the color contrast information that is lost in the luminance channel. A saturation-controlled adjustment of the output dynamic range is then adaptively performed to balance between the original range and the desired amount of enhancement.

Their algorithm achieves linear-time performance thanks to Gaussian pairing sampling which limits the amount of processed color differences. Three parameters are used to control contrast enhancement, scale selection and noise suppression, and image-independent default values for these parameters have been proposed.

3.2.5 Queiroz Invertible

Queiroz and Braun [9] have proposed an invertible conversion to grayscale. The idea is to transform colors into high frequency textures that are applied onto the gray image and can be later decoded back to color. The method is based on wavelet transformations and on the replacement of subbands by

[5]A simple color opponent space, see Section 2.1.3.f.

[6]Introduced in Section 2.1.5.

chrominance planes.

3.2.6 Alsam Sharpening

Alsam and Kolås [26] introduced a conversion method that aims to create sharp grayscale from the original color rather than enhancing the separation between colors. The approach resembles the Bala Spatial method (Section 3.2.1): first, a grayscale image is created by a global mapping to the image-dependent gray axis. Then, the grayscale image is enhanced by a correction mask in a way similar to unsharp masking (see Section 4.2 and reference [27]).

3.2.7 Neumann Adaptive

Recently, Neumann et al. [10] presents a local gradient-based technique with linear complexity that requires no user intervention.

It aims to obtain the best *perceptual* gray gradient equivalent by exploiting their Coloroid system and its experimental background. The gradient field is corrected using a gradient inconsistency correction method. Finally, a 2D integration yields the grayscale image.

In the same paper they also introduce another technique that is a generalization of the CIE LAB formula [21] (see Section 2.1.3.g): it introduces a signed power function to give a signum to the weighted Lab components that can be used as an alternative to the Coloroid gray gradient field.

3.2.8 Smith Apparent

A recent method by Smith et al. [11] combines global and local conversions in a way similar to the Alsam Sharpening method (see Section 3.2.6). The method first applies global "absolute" mapping based on the H-K effect (see

Section 3.1.2), and then locally enhances chrominance edges using adaptively weighted multi scale unsharp masking [28].

While the global mapping is image independent, the local enhancement reintroduces lost discontinuities only in regions that insufficiently represent the original chromatic contrast [11]. The goal of the method is perceptual accuracy, not the exaggeration of discriminability.

3.3 Gray4Matching: an ad-hoc idea

We present a new grayscale method based on the Grundland Decolorize algorithm (see Section 3.2.4) that is designed specifically for stereo and multi view stereo matching.

The basic idea is to transform the image set *preserving the consistence* between the images that are to be matched and in the meantime *optimizing* the transformation to preserve as much as possible of the original color information with respect to *the matching requirements*.

To achieve these objectives we must fulfill these requirements:

- *Feature Discriminability*: the method must preserve as much as possible of the image features discriminability to be matched, even at the cost of decreased *perceptual* accuracy of the image[7].
- *Chrominance Awareness*: the method must distinguish between isoluminant colors.
- *Color Consistency*: a color must be mapped to the same grayscale value in every image of the set to be matched.

[7]We will talk about the interesting correlations between perceptual and matching results in Section 5.4.

- *Global Mapping*: while the algorithm can use spatial information to determine the mapping, the same mapping must be used for every pixel of the image set.

- *Grayscale preservation*: if a pixel in the color image is already achromatic it should maintain the same gray level in the grayscale image.

- *Low Complexity*: if we consider the application of this algorithm in the prospect of multi view stereo matching, where a lot of images are to be processed, the computational complexity gains importance.

3.3.1 Analysis of the requirements

The choice of the starting point for the development of the Gray4Matching specification derives from an accurate analysis of the previous requirements. We examined every advanced technique from those listed in Section 3.2 and discarded the inapplicable ones for the theoretical reasons that follows.

Bala Spatial has been discarded because the spatial frequency based weighting of the importance of the H-K effect with respect to the base lightness violates the consistency and the globalist requisites. As already said, it is also susceptible to problems in chroma and lightness misalignment.

Gooch Color2Gray violates mainly the low complexity requirement: its $O(N^4)$ computational complexity is really too much for our context of application, and even Mantiuk's $O(N^2)$ improvement does not give us enough confidence with respect to quality versus complexity. Moreover there are issues with the algorithm's dependence on parameters that can arbitrarily affect the grayscale mapping: this can be good for artistic purposes but it is a serious issue for our objectives. At last, by applying gradient based minimization process it violates the color consistency, the globality and the grayscale preservation requirement.

Queiroz Invertible is totally inapplicable because is designed for “hiding” the color information in “invisible” parts of the grayscale image, this does not improve feature discriminability in any way with respect to the classical conversions.

Rasche Monochromats has problems regarding the tradeoff between complexity and quality of the results because it quantizes colors. Moreover it applies an energy minimization process that violates the color consistency, the globality and the grayscale preservation requirement.

Neumann Adaptive is not appropriate for matching because image details and salient features may be lost by unpredictable behavior in inconsistent regions of the gradient field. Another problem is that this approach is too much oriented toward *human* perception.

The main problem with Alsam Sharpening and Smith Apparent is that, like Bala’s approach, they violate our consistency and globality requisites because of their “unsharp masking”-like filtering of the images.

For these reasons, we develop an extension of the Grundland Decolorize, because it respects all our requisites. In Section 4.3.4 we will explain the original algorithm in more detail and in Section 4.3.5 we will explain how we adapted it for our goals.

3.4 Gamma compression and grayscale

Every step of an image processing should be applied over the linear representation of the image, not over the gamma precompensated one. In the real world this issue is ignored in the vast majority of the applications.

The main problem is that, often, we do not known anything about the image’s gamma.

Excluding Value HSV and RGB channel filter, that does not manipulate the color values but only choose one of them, the grayscale conversions considered in this chapter are all more or less affected by this issue.

The other basic techniques are relatively robust from this point of view, even if applying these conversion to gamma precompensated values make theoretically no sense.

It is difficult to predict the impact of this issue for the advanced techniques, even if we can speculate that Bala Spatial, Alsam Sharpening and Smith Apparent are the most robust ones, because they are basically weightings of the color values with spatial driven perturbations that enhance them. A study of the effects of this issue in approaches like Gooch Color2Gray, Rasche Monochromats, Neumann Adaptive and Queiroz invertible would be very complex and is completely out of the scope of this work, even if it is really interesting and we will probably make a dedicated study in future works.

Here we must say that Grundland Decolorize and (consequently) Gray4-Matching are very sensible to this issue, because they use saturation and the proportions between the image chromaticies to choose the mapping of a color hue to increases or decreases of the basic lightness. If the values are not linear, these ratios change significantly and the resulting mapping is heavily affected.

To highlight the importance of this issue, in Section 5.1 we repeat the same experiments with two version of our Gray4Matching: a version assumes that the images are stored in the sRGB format and consequently linearize the values prior to the computation, while the other version assume that the images are already linear.

Chapter 4

Implementations

From a development point of view, this work is not a standalone project but it is part of an ongoing effort in expanding the fields of interests of the Visual Computing Laboratory research group toward the Computer Vision[1].

As such, it was necessary to implement our grayscale conversions according to the laboratory's *software context*, explained in Section 4.1, building our system as an expansion of the *VCG Library* already used in our long term *PipeLab* project.

In Section 4.2 we will explain our generic image processing library module. In Section 4.3 we will describe the color to gray conversions that we choosed

[1]The other research areas of the Laboratory are *3D Scanning* (Low cost acquisition devices, improved acquisition methods, development of software tools for model construction, color acquisition and stitching), *Visualization Tools for Cultural Heritage* (Software tools for interactive visualization of 3D models, Polynomial Texture Maps), *Triangulated Surfaces Processing and Rendering* (Data structures and algorithms for surface management, surface simplification, multiresolution representation and visualization), *Deformable Object Modeling* (Multiresolution deformable object modeling), *Scientific Visualization* (Surface fitting techniques, Volume data processing), *Computational Geometry and Computer Graphics* (Basic geometric algorithms, Computational geometry, Computer graphics techniques).

to implement, including also some examples of conversion. Grundland Decolorize algorithm (Section 4.3.4) is explained in much more detail with respect to the other conversions because is the basis of our Gray4Matching optimization (Section 4.3.5).

We then focus on the stereo matching related work, describing the used framework in Section 4.4 and explaining the semi-automatic pre and post processing utilities that we built around it in Section 4.5.

4.1 Software context

The Visual Computing Laboratory research group has a well defined software context, that will be described in this section. The vast majority of the programming efforts are implemented in the *C++ language* [29, 30], together with some libraries such as *OpenGL* [31], *QT* [19] and others.

The Laboratory stresses the importance of good programming practices such as reutilization of the code, modularity and a uniform programming style. To achieve these objectives a single library has been built and is used for almost every project of the group.

VCGLib is a portable C++ templated library for manipulation and processing of triangle and tetrahedral meshes. This advanced mesh processing library is at the basis of all the Laboratory software tools. Composed by more than 50000 lines of code it currently includes a lot of algorithm for mesh processing task, like high quality quadric simplification, mesh cleaning, mesh smoothing and de-noising, file format parsing and so on.

The bigger and most widely used application of the Laboratory is *MeshLab* [32], that is a free and open-source general purpose mesh processing system aimed to help the management of the typical “not so small” unstructured 3D models that arise in the pipeline of processing of the data coming form 3D scanning activities in the context of Cultural Heritage. For this purposes MeshLab provides a set of tools for editing, cleaning, healing, inspecting, rendering and converting these kinds of meshes. The first stable version was downloaded 30000 times on a nine months period. The last version has been downloaded more than 5000 times in just the first three weeks. MeshLab serves as a single interface for most of the VCGLib functionality.

Our long term objective is to build *PipeLab*, a general multi view stereo matching framework, or, in other words, a modular *pipeline* for image based 3D reconstruction that can be viewed as MeshLab's equivalent high level program. To accomplish this goal we built two new library modules: one for image processing and one for computer vision, and we also already laid the core mechanisms of PipeLab. The grayscale conversions are implemented as functions of the computer vision module and both PipeLab and the preprocessing utilities of Section 4.5 use them.

Both MeshLab and the library are released under the GPL license and are available through anonymous svn. PipeLab will be released with the same license.

4.2 Implementation of the image processing module

An image processing module of VCGLib has been designed and built from scratch in order to support the grayscale conversion algorithms. This module has also been designed in order to substitute the ad-hoc small solutions developed for the various Laboratory projects[2].

The module is composed of a templated generic image class, an auxiliary system for management of the metadata attributes and a long series of separate functions.

[2]Until now a common image processing codebase was not necessary.

The image class is templated over three parameters:

- *The number of the image channels*: an image can have from one to an arbitrary large number of channels, i.e. values per pixel. The default is to have 3 color channels.
- *The type of the values*: the image stores the pixels' values in floating point variables. The default is to have double floating point precision, the other option is to have single floating point precision. Static assertions are used to ensure that the value type of an image is a floating point number.
- *The safeness*: the image throws runtime exceptions when its member functions are called with "wrong" parameters. This behavior, active by default, can be disabled in order to speed up the computation. The parameter correctness is also independently checked with the dynamic assertions mechanism that can be disabled by compiling the code in "release" mode.

The image data is packed by the pixel, to optimize multichannel processing efficiency. The image class interface is restricted to pixel and metadata access, processing and I/O are entirely delegated to external functions. The data is accessible in multiple ways: value and pixel wise, using float coordinates with nearest and bilinear interpolation, viewing the image with clamping, direct access to the data and so on.

An auxiliary structured attribute contains all the metadata information, currently this data consists in the specification of the numeric range of the values, the image color space, the white point and the gamma compression of the image. The various functions checks these data before processing, failing if the image is not compatible with the operation in course.

As the rest of VCGLib, the core of the library does not rely in other libraries, external wrapping modules have been implemented, for example, to implement the interactions with QT's platform independent I/O mechanisms.

We implemented many image processing functions that are "format safe" and that can operate efficiently. Between those there are normalization functions, color space transformations, color space aware gamma correction functions and many image filters such as a *Generic Convolution* filter (that can take a generic n by m convolution kernel), a *Mean* filter (noise reduction noise removal using mean of neighborhood), a *Gaussian Smoothing* filter (noise reduction using convolution with a Gaussian smoothing kernel), a *Laplacian* filter (edge detection filter), a *Laplacian of Gaussian* filter (edge detection filter), a *Difference of Gaussian* filter (a bandpass filter), a *Unsharp Mask* filter (edge enhancement) and a *Median* filter (noise reduction using median of neighborhood).

The various grayscale implementations described in Section 4.3 have *not* been included in this module but in a preprocessing submodule of the computer vision module, a default CIE Y implementation has also been included here for basic RGB to grayscale conversions.

The image processing module is designed to be used inside other projects, for example to support multispectral textures in MeshLab and High Dynamic Range images in other Laboratory projects.

4.3 Grayscale conversion implementations

After the theoretical review of the state of the art in color to gray conversion that we explained in Chapter 3, we had to choose the subset of the algorithms that had to be implemented and tested in the matching context.

We implemented all these conversions in the preprocessing submodule of the new computer vision module in the VCGLib[3].

We decided to first implement almost all of the basic techniques of Section 3.1 as a quick way to get a preliminary grasp of the practical effects of different conversions in stereo matching. These preliminary results were not relevant to the scopes of our work and all but one of them have been excluded from the results of Chapter 5. All the implementations remains in the submodule, ready to be used to test their impact in future multi view stereo matching works.

We implemented and tested three advanced techniques between the eight described in Section 3.2 for the following motivations:

- Queiroz Invertible has been immediately discarded because, as we already said in Section 3.3.1, does not improve feature discriminability in any way with respect to the classical conversions.
- Rasche Monochromats and Neumann Adaptive have been discarded respectively for the color quantization problem and for the unpredictable behavior in inconsistent regions of the gradient field.
- between the three similar techniques Bala Spatial, Alsam Sharpening and Smith Apparent we choosed to implement the most recent one, Smith Apparent.
- Gooch Color2Gray has been implemented in order to demonstrate how its computationally heavier complexity does not improve in practice the quality of the results, at least in our context.
- Grundland Decolorize has been implemented in order to show the differences between its performance and Gray4Matching's one.

[3]The other implemented submodules are not relevant to this work's objectives and will not be explained here.

We finally implemented our Gray4Matching extension of the Grundland Decolorize conversion.

4.3.1 Implementation of the basic methods

We quickly implemented functions for Value HSV, RGB Channel Filter, Lightness HSL, Naive Mean, NTSC Rec.601, CIE Y and Lightness Nayatani VAC methods. The QT qGray was already implemented, and even if it is possible to implement Fairchild Lightness and Lightness Nayatani VCC we found that it was not necessary.

There is not much to say about these implementation over what we already explained in Section 3.1.

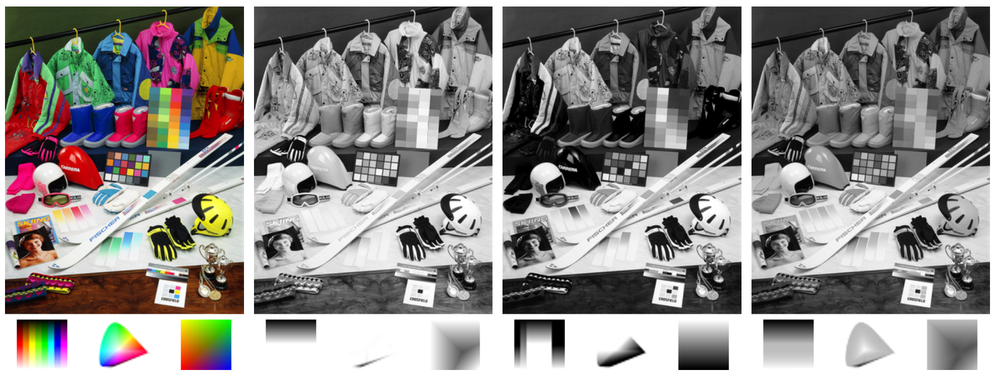
As can be easily seen in the example provided in Figure 4.1, the first three conversions ((b), (c) and (d)) are to be avoided in every occasion because they discards too much informations and loose features, channel averaging (e) begins to give “acceptable” results and there are not many noticeable differences between the last three cases ((f), (g) and (h)). In Figure 3.1 at page 26 the difference between Lightness Nayatani VAC and CIE Y can be seen much more evidently.

In the following, only CIE Y is used in the experimental results.

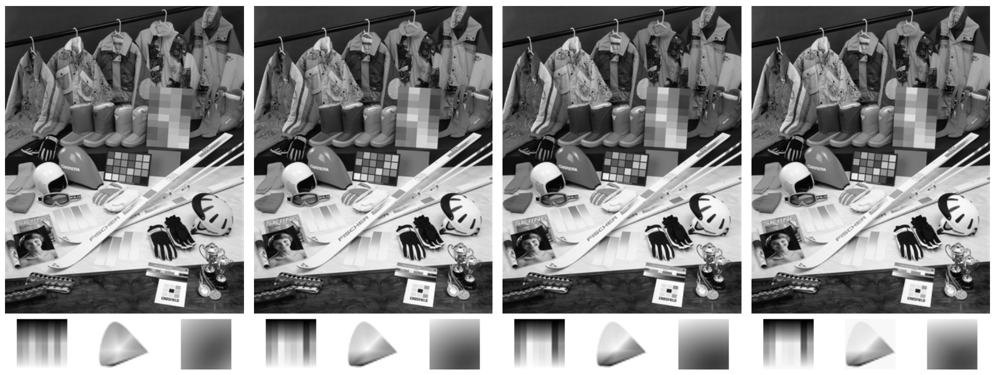
4.3.2 Implementation of Gooch Color2Gray

The Gooch Color2Gray algorithm is composed by three steps:

1. The color image is converted in a perceptually uniform color space.
2. Target differences are computed in order to combine luminance and chrominance differences.
3. A least squares optimization is used to selectively modulate the source



(a) Original colors **(b)** Value HSV **(c)** Green Filter **(d)** Lightness HSL



(e) Naive Mean **(f)** NTSC Rec.601 **(g)** CIE Y **(h)** Nayatani VAC

Figure 4.1: *An example of the implemented basic techniques.*

luminance differences in order to reflect changes in the source images chrominance.

The first step takes a linear RGB image and converts it in the CIE LAB representation. See Section 2.1.3.c and Section 2.1.3.g for details about these spaces.

The color differences between pixels in the color image are expressed as a set of signed scalar values. For each pixel i and neighbor pixel j a signed distance scalar δ_{ij} based upon luminance and chrominance differences between i and j is found. Using these values, g values for the grayscale image are then derived. The grayscale difference between pixel i and a neighboring pixel j is $(g_i - g_j)$.

The conversion objective consists of finding g such that all $(g_i - g_j)$ values closely match the corresponding δ_{ij} values. Specifying δ_{ij} is fairly involved, and user interaction is needed in order to obtain acceptable results.

The output image g is found by an iterative optimization process: given a set of desired signed differences δ_{ij}, the gray image g is found by minimizing the following objective function, $f(g)$, where K is a set of ordered pixel pairs (i, j):

$$f(g) = \sum_{(i,j)\in K} ((g_i - g_j) - \delta_{ij})^2 \tag{4.1}$$

Where g is initialized to be the luminance channel of the source image, and then descends to a minimum using conjugate gradient iterations [33]. In order to choose a single solution from the infinite set of optimal g, the solution is shifted until it minimizes the sum of squared differences from the source luminance values.

The algorithm's behavior heavily relies on the choice of the three user parameters:

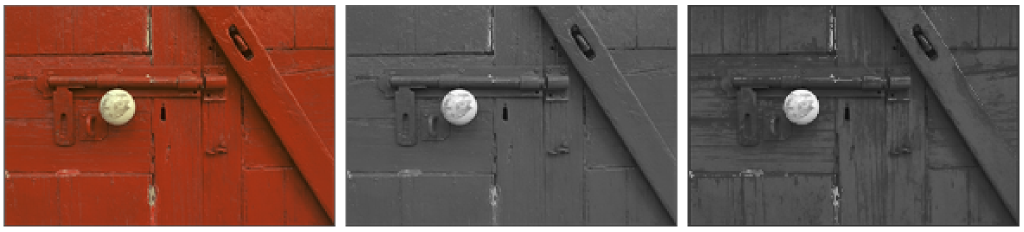
- θ controls whether chromatic differences are mapped to increases or decreases in luminance value.
- α determines how much chromatic variation is allowed to change the source luminance value.
- μ sets the neighborhood size used for chrominance estimation and luminance gradients, a 0 value means full neighborhood.

As already said in Section 3.2.2, the algorithm has a $O(N^4)$ computational complexity, when a full neighborhood is used ($\mu = 0$). Mantiuk's approach [25] can speed up the computation using a multiresolution framework but, given the "experimental" approach of this work, we implemented the original version.

The implementation, executed on a Intel Core 2 Duo T7100 @ 1.80 GHz laptop with 2GB of ram and a Gentoo Linux operative system takes:

- approximately 11 minutes for a 256×256 image.
- approximately 3 hours for a 512×512 image.
- approximately 22 hours for a 1024×1024 image.

For an example of the conversion, Figure 4.2 shows a comparison between Color2Gray and CIE Y on a small image, where can be noticed that Gooch's approach (c) strongly overemphasizes the little details of the wood texture with respect to both the original image (a) and the classical method (b).



(a) Original colors **(b)** CIE Y **(c)** Gooch Color2Gray

Figure 4.2: *An example of a Gooch Color2Gray conversion with a CIE Y reference on a 192×128 image. Gooch's parameter are $\theta = 45$, $\alpha = 10$ and $\mu = 0$ (full neighborhood), the conversion took 106.5 seconds for Color2Gray, and 0.002 seconds for CIE Y.*

4.3.3 Implementation of Smith Apparent

The Smith Apparent algorithm can be summarized by the following two steps:

1. The color image is converted in grayscale using the Lightness Nayatani VAC technique.
2. The image contrast is enhanced using unsharp masking adaptively weighted using the chrominance information.

The first step takes a linear RGB image and converts it in the CIE LUV representation. See Section 2.1.3.c and Section 2.1.3.h for details about these spaces.

The lightness channel is then altered according to the H-K effect (see Section 3.1.3) according to Formula 3.9.

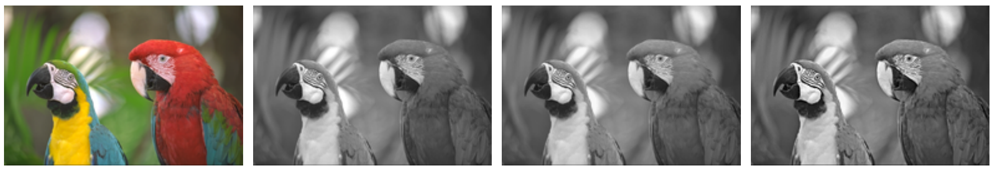
To counter the reduction of local contrast in the grayscale image, unsharp masking is used to better represent the local contrast of the original color image. At this point our implementation slightly differ from the technique described in [11]: while they use a general adaptively weighted multi scale unsharp masking technique [28], we make a simplification and use a single scale unsharp masking. The technique is adapted according to the ratio between color and grayscale contrast, so that increases occur at under-represented color edges without unnecessarily enhancing edges that already represent the original.

For an example of the conversion, Figure 4.3 shows a comparison between Smith Apparent, Lightness Nayatani VAC and CIE Y on a colorful image , where can be noticed how Nayatani VAC (c) improves over CIE Y (b) in the hue change of the red parrot's wing and how Smith Apparent (d) restores the details of the image almost to the original quality (a).

4.3.4 Implementation of Grundland Decolorize

The Grundland Decolorize algorithm has five steps:

1. The color image is converted in a color opponent color space.



(a) Original colors **(b)** CIE Y **(c)** Nayatani VAC **(d)** Smith Apparent

Figure 4.3: *An example of a Smith Apparent conversion, compared to CIE Y and to the algorithm's intermediate step Lightness Nayatani VAC.*

2. The color differences are measured with Gaussian sampling.
3. The chrominance projection axis is found by predominant component analysis
4. Luminance and chrominance information are merged.
5. The dynamic range is adjusted using the saturation information.

The process is controlled by three parameters: the degree of image enhancement, λ, the typical size of relevant image features in pixels, σ, and the proportion of image pixels assumed to be outliers, η. Defaults values are $\lambda = 0.5$, $\sigma = 25$ and $\eta = 0.001$.

The first step takes a linear RGB image (with values in $[0..1]$ range) and converts it in their YPQ representation.

The YPQ color space consists in a luminance channel Y and two color opponent channels: a yellow-blue P and a red-green Q. The formula for Y is the NTSC Rec.601 one (see Section 3.1.2.b), while $P = \frac{R+G}{2} - B$ and $Q = R - G$. The perpendicular chromatic axes support easy calculation of hue $H = \frac{1}{\pi}\tan^{-1}\left(\frac{Q}{P}\right)$ and saturation $S = \sqrt{P^2 + Q^2}$.

To analyze the distribution of color contrasts between image features, color differences between pixels are considered. The algorithm uses a novel randomized scheme, sampling by Gaussian pairing.

Each image pixel is paired with a pixel chosen randomly according to a displacement vector from an isotropic bivariate Gaussian distribution. The horizontal and vertical components of the displacement are each drawn from a univariate Gaussian distribution with mean 0 and variance $\frac{2}{\pi}\sigma$. The displacement is carried out under symmetric boundary conditions. Each pixel contributes to at least one color contrast. Repeatable results are guaranteed by initializing the random number generator with a fixed seed[4].

To find the color axis that represents the chromatic contrasts lost when the luminance channel supplies the color to grayscale mapping, predominant component analysis is used. In the PQ chrominance plane, the predominant axis of chromatic contrast is determined through a weighted sum of the oriented chromatic contrasts of the paired pixels. The weighs are determined by the *contrast loss ratio*[5] and the ordering of the luminance.

Unlike principal component analysis which optimizes the *variability* of observations, predominant component analysis optimizes the *differences* between observations. The predominant chromatic axis aims to capture the color contrast information that is lost in the luminance channel. The direction of the predominant chromatic axis maximizes the covariance between chromatic contrasts and the weighted polarity of the luminance contrasts.

Next, luminance and chrominance information are combined. The predominant chromatic data values are obtained by projecting the chromatic data onto the predominant chromatic axis.

To appropriately scale the dynamic range of the predominant chromatic channel the algorithm ignores the extreme values due to the level η of image

[4]We use the improved Marsenne Twister uniform generator [34] and then use the polar form of the Box-Muller transformation [35] to obtain a Gaussian distribution. Our fixed seed is 42, obviously.

[5]The relative loss of contrast incurred when luminance differences are used to represent the RGB color differences.

noise. To detect outliers, a linear time selection algorithm is used to calculate the outlying quantiles of the image data. It is applied to discount the extreme absolute values of the projected chromatic data when scaling the predominant chromatic channel.

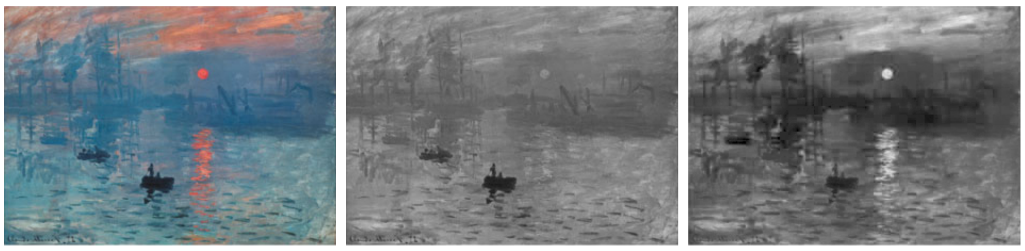
The predominant chromatic channel is combined with the luminance channel to produce the desired degree λ of contrast enhancement. At this intermediate stage of processing, the enhanced luminance is an image dependent linear combination of the original color, mapping linear color gradients to linear luminance gradients.

The final step uses saturation to calibrate luminance while adjusting its dynamic range and compensating for image noise. When the dynamic range of the enhanced luminance is evaluated, the effects of image noise are excluded. The corrected dynamic range expands the luminance channels original dynamic range to accommodate the desired degree λ of contrast enhancement.

The corrected luminance is obtained by linearly rescaling the enhanced luminance to fit the corrected dynamic range. Saturation channel is used to derive the bounds on the distortion permitted in achieving the degree λ of contrast enhancement. To ensure that achromatic pixels retain their luminance as their gray level, the discrepancy between luminance and gray level needs to be suitably bounded. The output gray levels are obtained by clipping the corrected luminance to conform to the saturation dependent bounds.

The resulting transformation to gray levels is a continuous, piecewise linear mapping of color and saturation values. To produce its output image, the algorithm linearly combines the luminance channel with feedback from either the predominant chromatic channel or the saturation channel. Hence, the enhanced contrast originates from either a linear or a polar representation of chromatic data. In both cases, the polarity of the predominant chromatic channel serves to determine whether the feedback is positive or negative. Moreover, the predominant chromatic channel acts as a bridge between positive and negative saturation feedback. The contrast enhancement parameter moderates the predominant chromatic channel feedback, the saturation channel feedback, and the dynamic range expansion.

For an example of the conversion, Figure 4.4 shows a comparison between Grundland Decolorize and CIE Y on a “difficult” image, where can be noticed that Grundland’s approach (c) strongly improves feature discriminability with respect to the classical method (b), restoring almost every feature of the color image (a).

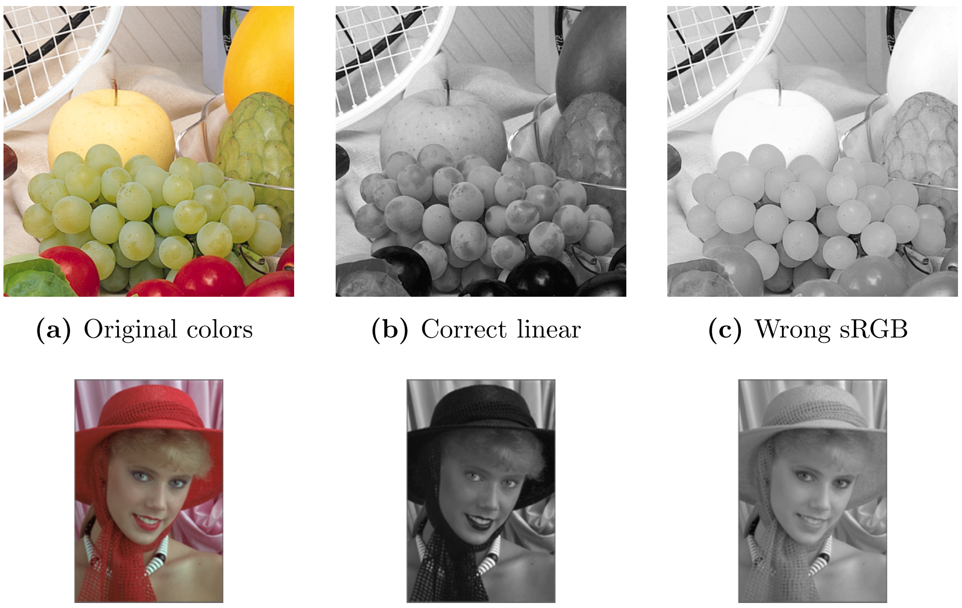


(a) Original colors **(b)** CIE Y **(c)** Grundland Decolorize

Figure 4.4: *An example of a Grundland Decolorize conversion with a CIE Y reference.*

As already mentioned in Section 3.4, Grundland Decolorize is really sensible to the issue of gamma compression. In Figure 4.5 we show two examples of how a wrong gamma assumption can decrease the quality of the results. A color image (a) has been linearized and then converted correctly assuming linearity (b) and wrongly assuming sRGB gamma compression (c). To show the complementary case, an sRGB image (d) has been converted wrongly assuming linearity (e) and correctly assuming its gamma compression (f). The loss of information is evident in the conversion that makes the wrong assumption: light areas (c) or dark areas (e) loses most of the features because the saturation balancing interacts wrongly with the outliers detection.

Moreover, the predominant chromatic axis is perturbed and consequently the chromatic projection does not retain its original meaning anymore, see for example how the red hat and the pink skin (d), that should be mapped to similar gray intensities (f), are instead mapped to very different intensities (e).



(a) Original colors **(b)** Correct linear **(c)** Wrong sRGB

(d) Original colors **(e)** Wrong linear **(f)** Correct sRGB

Figure 4.5: *Two examples of right and wrong gamma assumptions with Grundland Decolorize.*

4.3.5 Implementation of Gray4Matching

As said before, Gray4Matching is an adaptation of the Grundland Decolorize algorithm that evaluates the whole set of images to be matched simultaneously. To achieve this, we modified our implementation of Grundland's algorithm in order to execute each of the five steps simultaneously for every image of the set.

Excluding a factor, this seems equivalent to the following procedure:

1. Stitch together, side by side, all the images of the set in order to make a single big image.
2. Compute the Grundland Decolorize algorithm on the “stitched” image.
3. Cut back the grayscale version of the original images.

Nevertheless, this hypothetical implementation would not work correctly because, in the Gaussian sampling step, near the common borders of the images would occur that a pixel could be paired with a pixel near the border of another image and the color differences estimation would be altered.

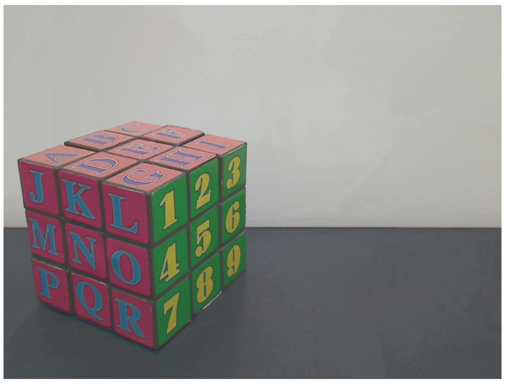
The original requirements of our conversion, explained in Section 3.3, were *Feature Discriminability*, *Chrominance Awareness*, *Color Consistency*, *Global Mapping*, *Grayscale preservation* and *Low Complexity.* With Gray4Matching these advantages of the Grundland Decolorize algorithm are applied *consistently* in the set of the images. Moreover, the results can benefit from the following proprieties of the transformation:

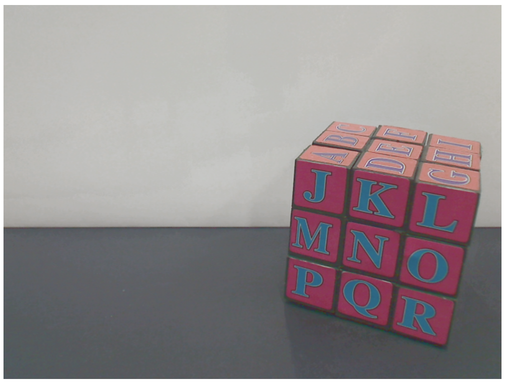
- *Contrast Magnitude*: the magnitude of the grayscale contrasts visibly reflects the magnitude of the color contrasts.
- *Contrast Polarity*: the positive or negative polarity of gray level change in the grayscale contrasts visibly corresponds to the polarity of luminance change in the color contrasts.
- *Dynamic Range*: the dynamic range of the gray levels in the grayscale image visibly accords with the dynamic range of luminance values in the color image.
- *Continuous mapping*: the transformation from color to grayscale is a continuous function. This reduces image artifacts, such as false contours in homogeneous image regions.

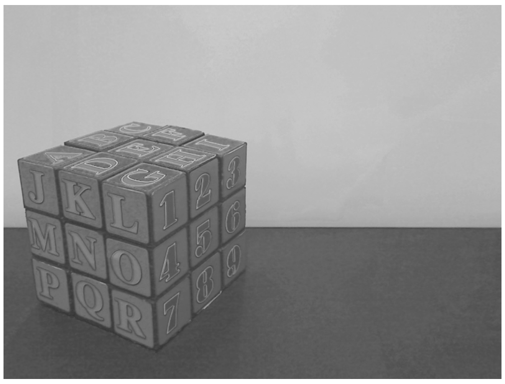
- *Luminance ordering*: when a sequence of pixels of increasing luminance in the color image share the same hue and saturation, they will have increasing gray levels in the grayscale image. This reduces image artifacts, such as local reversals of image polarity.
- *Saturation ordering*: when a sequence of pixels having the same luminance and hue in the color image has a monotonic sequence of saturation values, its sequence of gray levels in the grayscale image will be a concatenation of at most two monotonic sequences.
- *Hue ordering*: when a sequence of pixels having the same luminance and saturation in the color image has a monotonic sequence of hue angles that lie on the same half of the color circle, its sequence of gray levels in the grayscale image will be a concatenation of at most two monotonic sequences.

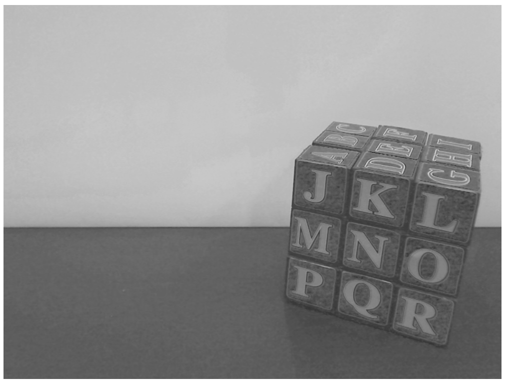
In Figure 4.6 we show how our approach improves over the original one in the stereo matching context. An example of two stereo image pair is shown. While Grundland's approach gives better results when considering the images separately, its results are completely inappropriate when the pair of images is considered together. For example see the "L–G–1" corner of the cube:

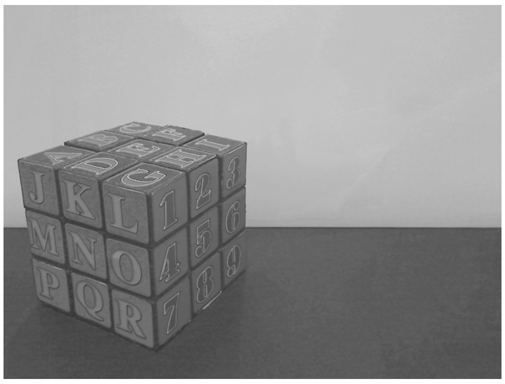
- In the "right" image (a), both Grundland (c) and Gray4Matching (e) have to cope with the presence of the green "1" side, and they obtain similar results.
- In the "left" image (b), where the green "1" side does not appear, Grundland (d) better distinguishes the background of the "L" from the letter color with respect to Gray4Matching (f).
- If the "left" and "right" images were to be matched, the vast majority of the algorithms would simply fail in matching the Grundland pair


(a) Original colors “right”


(b) Original colors “left”


(c) Grundland Decolorize “right”


(d) Grundland Decolorize “left”


(e) Gray4Matching “right”

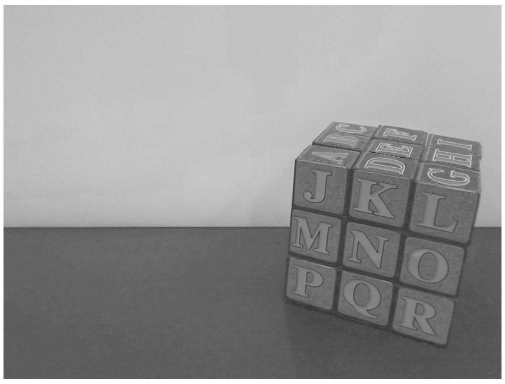

(f) Gray4Matching “left”

Figure 4.6: *Difference between Gray4Matching and Grundland Decolorize in a stereo pair where chrominance changes significantly.*

(c) and (d), while the Gray4Matching pair (e) and (f) has an high probability to give good matching results.

Please note that this example has been made to emphasize the differences of the two approaches and better explain the advantages of our adaptation, in real life scenarios these situation occurs in a softer way, at least in stereo matching. In multi view stereo matching, where more images are involved, the benefits of a consistent mapping will be much more relevant even in standard scenarios.

As Grundland Decolorize, Gray4Matching is sensible to alterations of the image gamma and, as such, knowledge of the starting image encoding is really necessary.

This first Gray4Matching implementation is just a proof of concept and issues of efficiency and scalability have not been considered. For example, while the algorithm has linear computational complexity, the memory occupation requirements of the current implementation are relatively heavy and does not scale well in the number and in the size of the images. These issues will be resolved in future implementations, as explained in Section 6.1.

4.4 The StereoMatcher framework

Stereo matching is one of the most active research areas in computer vision. While a large number of algorithms for stereo correspondences estimation have been developed, relatively little work has been done on characterizing their performance until 2002, when Scharstein and Szeliski presented a taxonomy paper [14] of dense two frame stereo methods.

Their taxonomy is designed to assess the different components and design decisions made in individual stereo algorithms. Using this taxonomy, they compare existing stereo methods and present experiments evaluating the performance of many different variants. In order to establish a common software platform and a collection of data sets for evaluation, they designed StereoMatcher, a standalone, flexible C++ software that enables the evaluation of individual components and that can easily be extended to include new algorithms. They also produced several new multi frame stereo data sets with groundtruth and made both the code and the datasets available on the Web. They concludes the taxonomy paper with a comparative evaluation of a large set of the best performing stereo algorithms that were available in 2002.

To test the impact of the different color to gray conversions in the stereo matching results, we used their widely known StereoMatcher.

The implementation is closely tied to the taxonomy presented in their paper and includes window-based algorithms, diffusion algorithms, as well as global optimization methods using dynamic programming, simulated annealing, and graph cuts. While many published methods include special features and post processing steps to improve the results, StereoMatcher implements the basic versions of such algorithms, in order to assess their respective merits most directly[6]. The implementation is modular and can be extended to include other algorithms or their components. The implementation contains a mechanism for specifying parameter values that supports recursive script files for performance comparisons on multiple data sets. Once a script is parsed and a dataset has been loaded the four steps that we described in Section 2.2.1 are (optionally) executed in sequence, and the performance quality evaluator is then invoked.

For the above reasons StereoMatcher is not an implementation of “state of the art” stereo methods, but includes many widely diffused standard algorithms.

[6]Please note that they exclude *post* processing steps and special features, their color treatment is the “normal” one.

Regardless of the algorithmic combination selected by the script, the treatment of the color is executed in the first step, the matching cost computation one. In the taxonomy paper, the matching cost computation is explained in the following way:

> *"The simplest possible matching cost is the squared or absolute difference in color / intensity between corresponding pixels. To approximate the effect of a robust matching score [36, 37], we truncate the matching score to a maximal value. When color images are being compared, we sum the squared or absolute intensity difference in each channel before applying the clipping. If fractional disparity evaluation is being performed, each scanline is first interpolated up using either a linear or cubic interpolation filter [38]. We also optionally apply Bircheld and Tomasi's sampling insensitive interval-based matching criterion [39], i.e., we take the minimum of the pixel matching score and the score at $\pm\frac{1}{2}$-step displacements, or 0 if there is a sign change in either interval. We apply this criterion separately to each color channel, which is not physically plausible (the sub-pixel shift must be consistent across channels), but is easier to implement."*

While this treatment has the advantage of using the color information, it is inappropriate from our point of view, because when a color image is given it blindly sums the absolute or the squared differences. Moreover, when the sampling insensitive matching criterion is used, it introduces relevant inconsistencies in the theoretical model.

While this situation may seem subpar, it is representative of the underestimation of the color treatment's importance in existing standard matching techniques, and strengthen our conviction that the *separation* of color treatment and matching cost computation is the best practice.

Strong of this conviction we did not modify their code but we built a preprocessing program, explained in Section 4.5, and we used the resulting grayscale datasets as inputs for StereoMatcher. As can be seen in the results, our approach often *improves* over the original one.

4.5 Pre and post processing of the StereoMatcher datasets

For our experiments we used many dataset with groundtruths mainly provided by the StereoMatcher authors [40, 41, 42, 43]. A first "preprocessing" step had to be the manual conversion of the datasets they published in the years after 2002 to a format usable by their own StereoMatcher[7].

We then preprocess the color datasets in two semiautomatic steps:

1. A step that duplicates the color dataset in the various grayscale copies and adapts the dataset parameters accordingly.

2. A step that creates the sequence of scripts that are needed in order to configure and control the StereoMatcher algorithms

The first step is configured with the list of the datasets that are to be converted and the list of conversion that are to be executed. Because of the large number of images that had to be converted and the presence of slow conversions[8], we implemented a dispatcher/worker parallel process system, able to take advantage of the modern multicore processor parallel features. Another expedient that really speeded up the experiments has been to automatically skip a conversion if it was already present in the destination directory. Per-dataset input configuration files are modified as needed.

[7]They probably use an unpublished newer version of their software and publish the datasets in the newer format.

[8]Gooch Color2Gray, in particular.

The second step takes the list of StereoMatcher algorithms that are to be executed and automatically generates the hundreds of ad-hoc StereoMatcher scripts needed to perform the evaluations. It also groups them in organized batches in order to better distribute the workload in multiple machines. The complexity of some iterative stereo matching algorithms made the experimentation times really long, requiring in total over a month of computations over four different machines[9].

After the experimental phase, we implemented and used an automatic post processing tool in order to:

- Extract the relevant input images, the groundtruths and the computed disparity maps from the dataset.
- Extract the numerical data from the output dataset files and rearrange it in order to compare the results of the different grayscale conversions.
- Generate and execute scripts for the *Gnuplot* tool that we used to generate the histograms of the rearranged numerical results shown in Chapter 5

[9]As explained in Chapter 5, we made a lot of experiments with different algorithms, different conversions and different dataset that are not included here for space reasons.

Chapter 5

Results

This Chapter will describe and discuss the results of the experimental evaluation of the proposed algorithm for grayscale conversion when it is applied to the stereo matching context. We will show how the choice of the preprocessed algorithms can improve the precision in the reconstruction of a depth map from a single stereo pair, independently from the reconstruction algorithm. In Section 5.1, the settings of the performed experiments are explained, and in Section 5.2 the results are commented. A relatively small selection of the StereoMatcher obtained results is shown in more detail in Section 5.3.

In Section 5.4 we will also compare the observed results in this field of application with a recent study [44] of the perceptual performances of the various grayscale conversions.

5.1 The experiments

To evaluate in a thorough way how the choice of different grayscale conversions affects the results computed by the StereoMatcher algorithms we have performed a large battery of tests. More than 2300 error measures has been taken in total, crossing different grayscale conversions with different

StereoMatcher algorithms and with different error measures.

To describe the experiments in an orderly fashion, we can catalog them according to the following classification:

1. *The dataset*: we used twelve different datasets with groundtruth, these dataset are some of the standard ones used in the Computer Vision community.

2. *The StereoMatcher algorithmic combination*: we used three different standard algorithms to obtain the depth maps.

3. *The class of error measure*: we used two different kinds of measures of the computed depth maps errors.

4. *The area of interest of the error measure*: we measured the errors in four differently characterized "areas" of the depth maps.

5. *The grayscale preconversion*: we used both the original color datasets, two versions of our proposed grayscale conversion and five other different grayscale conversions.

This classification facilitates the comparison of the various relative advantages and disadvantages of the grayscale conversions with respect to both the StereoMatcher algorithms and the peculiarities of the datasets. We will explain each of these elements in more detail in the following Sections.

5.1.1 The datasets

The used dataset comes mainly from many subsequent works of the StereoMatcher authors. Except one dataset, proposed in [40] and redistributed by them, they were proposed in subsequent years [41, 42, 43] and they are commonly used in the Computer Vision community.

- The 1996 “tsukuba” dataset.
- Three 2001 datasets: “sawtooth”, “venus” and “map”
- Three 2005 datasets: “dolls”, “laundry” and “reindeer”.
- Three 2006 datasets: “aloe”, “cloth” and “plastic”.
- Two variations of the datasets “aloe” and “dolls” where the illumination significantly changes between the left and the right images.

The “map” dataset is originally in grayscale and has been used in order to validate our theoretical prevision that our conversion preserves the image quality when the colors are already achromatic.

For obvious space reasons we report only the results relative to four datasets: “tsukuba” in Section 5.3.1, “aloe” in Section 5.3.2, “dolls” in Section 5.3.3 and “laundry” in Section 5.3.4. Partial results and the motivations for the illuminant variations experiments that use the modified versions of the “aloe” and the “dolls” datasets are given in Section 5.3.5

5.1.2 The StereoMatcher algorithmic combinations

As explained in Section 2.2.1, the stereo matching process takes two rectified images of a three dimensional scene and computes a disparity map, an image that represents the relative shift of the scene features between the images. The magnitude of this shift is inversely proportional to the distance between the observer and the feature. In the experiments, to obtain the computed depth maps we used the following StereoMatcher algorithmic combinations:

- `WTA-AD`: a *Winner Take All* disparity computation with *Absolute Differences* matching cost computation.
- `WTA-SD`: a *Winner Take All* disparity computation with *Squared Differences* matching cost computation.

- `GC-AD`: a *Graph Cuts* disparity computation with *Absolute Differences* matching cost computation.

The *Winner Take All* disparity computation algorithm simply picks the lowest matching cost as the selected disparity at each pixel. The *Graph Cuts* disparity computation is an iterative energy minimization algorithm that tries to enhance the *smoothness term* of the computed disparity maps. Explaining in detail these algorithms is out the scope of this work, see [45] for the *Graph Cuts* algorithm and [14] for both the used algorithm and the other StereoMatcher implementations. As already explained in Section 4.4, the *Absolute Differences* matching cost computation simply sums the absolute RGB differences between two pixels, while *Squared Differences* sums the squared RGB differences. Both cost computations truncate the sum to a maximal value in order to approximate the effect of a robust matching score. For every algorithm we use a fixed aggregation window (the spatial neighborhood considered in the matching of a pixel) and no sub-pixel refinements of the disparities[1].

5.1.3 The classes of error measures

To evaluate the performance of the various grayscale preconversions, we need a quantitative way to estimate the quality of the computed correspondences. A general approach to this is to compute error statistics with respect to the groundtruth data. The current version of StereoMatcher computes the following two quality measures based on known groundtruth data:

- `rms-error`: *root-mean-squared error* (measured in disparity units) between the computed disparity map $d_C(x, y)$ and the groundtruth map

[1]Again, see [14] for the details of the StereoMatcher algorithms.

$d_T(x,y)$, i.e.,

$$R = \left(\frac{1}{N} \sum_{(x,y)} |d_C(x,y) - d_T(x,y)|^2 \right)^{\frac{1}{2}} \tag{5.1}$$

where N is the total number of pixels.

- `bad-pixels`: the percentage of bad matching pixels:

$$B = \frac{1}{N} \sum_{(x,y)} (|d_C(x,y) - d_T(x,y) > 1|) \tag{5.2}$$

5.1.4 The areas of interest of the error measures

In addition to computing the statistics over the whole image, we also focus on three different kinds of regions. These regions are computed by preprocessing the reference image and the groundtruth disparity map to yield the following three binary segmentations:

- *textureless regions*: regions where the squared horizontal intensity gradient averaged over a square window of a given size is below a given threshold;
- *occluded regions*: regions that are occluded in the matching image, i.e., where the forward-mapped disparity lands at a location with a larger (nearer) disparity;
- *depth discontinuity regions*: pixels whose neighboring disparities differ by more than a predetermined gap, dilated by a window of predetermined width.

These regions were selected to support the analysis of matching results in typical problem areas.

We report the measures for the whole image (`all`), for the regions that are *not* occluded (`nonocc`), for the textureless regions (`textureless`) and for the depth discontinuity regions (`discont`).

5.1.5 The grayscale preconversion

We executed the StereoMatcher algorithms and measured the various error measures for the following versions of every dataset:

- Original color version, because we obviously needed a starting point to understand if our conversions would give worse, equal or even better results with respect to the standard color approach.
- CIE Y has been choosed as the representative of the standard grayscale conversions.
- Gooch Color2Gray, Grundland Decolorize and Smith Apparent have been included.
- Gray4Matching have been included in two flavors: a version that assumes linearity of the input images and a version that assumes sRGB gamma compression.

5.2 Comment on the results

From the observations of the experiment results we can draw some conclusions, summarized in this section.

We can observe that the pre-2002 StereoMatcher algorithms cannot cope with "difficult" datasets like most of the 2005 and the 2006 series. However this not invalidates our experiments, because it is important to assert the impact of different grayscale conversions in these widely used standard algorithms.

From the results relative to the grayscale "map" dataset we can confirm our prevision of Section 3.3.1 that only Gooch Color2Gray damages the images that are already in grayscale.

The experiments with the change of luminance shows that, as expected, no grayscale preprocessing step can recover from the incapacity of the standard matching algorithms when extreme changes of luminance are present between the images (see Section 5.3.5 for more details).

We have no official informations about the gamma encoding of our dataset but, by means of empirical measures of the images histogram distributions, we deduced that only the datasets from 2006 are gamma compressed. Comparisons between the results of the linear and the sRGB versions of our Gray4Matching conversion seem to confirm this hypothesis.

StereoMatcher's *standard* approach to the color information is often inappropriate: the original color versions frequently gives equal or worsen results with respect to the "good" grayscale conversions.

The grayscale conversions performances can be summarized with the following statements:

- CIE Y has a good ratio between complexity and performance, in general the other conversion are better.
- Gooch Color2Gray gives bad results, especially when considering its computational complexity.
- Grundland Decolorize has varied results, but it is constantly worse than Gray4Matching.
- Smith Apparent gives interesting results, its chrominance driven unsharp masking gives results that are often equal or better with respect to Gray4Matching.
- The "right" gamma version of Gray4Matching is constantly one of the best conversions, together with Smith Apparent and the original version.

5.3 Detailed samples of the StereoMatcher results

In this section we show a (relatively) small selection of the experimental datasets, with full details of four datasets and a summary of the luminance experiment.

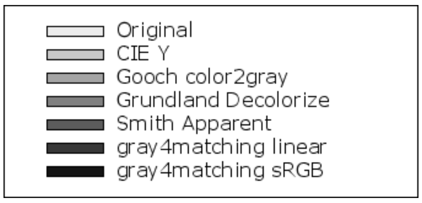


Figure 5.1: *The legend of the histograms.*

For every dataset, first a brief explanation is given, then the disparity map (DM) groundtruth is shown, followed by the display of the stereo pair and the three computed disparity maps for every color conversion, lastly the histograms of the error measures are given. Every histogram follows the color scheme of Figure 5.1 and, for each class of errors, the same scale is used consistently between the datasets in order to give an idea of the overall difficulty of the various datasets. As already explained, every dataset has an intrinsic scene complexity and the most recent ones, like “aloe”, are more difficult with respect to the older ones, like “tsukuba”.

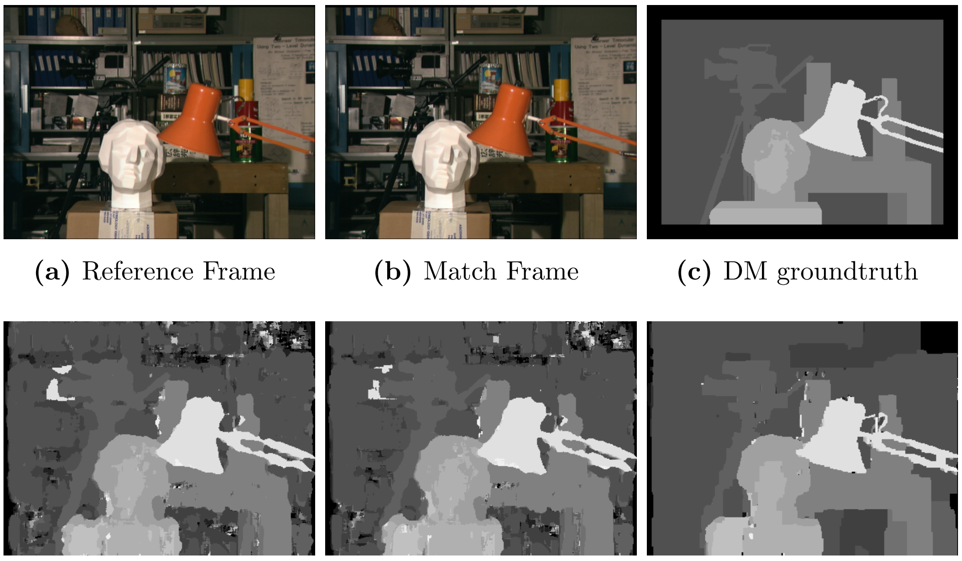
5.3.1 Results for dataset “tsukuba”

The “tsukuba” dataset is one of the first datasets made that is equipped with groundtruth informations. Is also one of the most widely used in the Computer Vision community.

For this dataset every used algorithm works fine with every conversion, because this depicts a really simple scene, mainly composed by planar and almost planar pieces. The most difficult feature of the dataset, when working in grayscale, is the lamp support beams, that are almost isoluminant with respect to the background.

When the `GC-AD` algorithm is used, our conversion is the only one that obtains the same quality of the original colors.

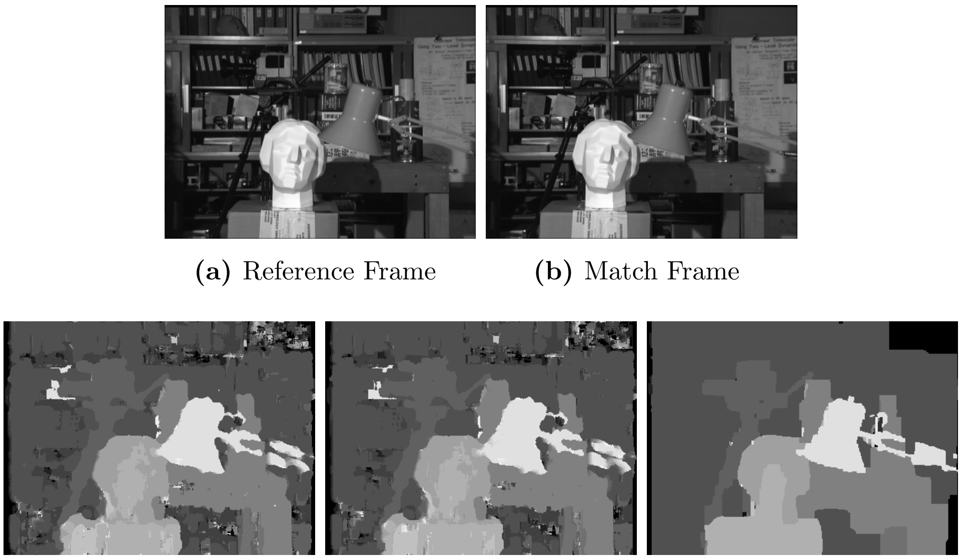
For a gamma compression point of view this dataset seems to be already linear, but there is only this dataset from 1994 and so we cannot be confident of this hypothesis as with the other datasets.



(a) Reference Frame **(b)** Match Frame **(c)** DM groundtruth

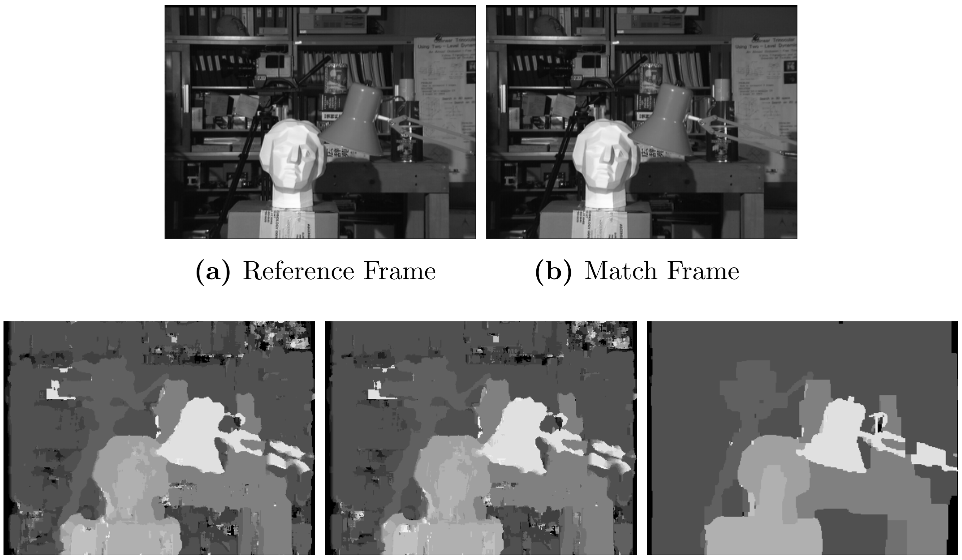
(d) DM from WTA-AD **(e)** DM from WTA-SD **(f)** DM from GC-AD

Figure 5.2: *“tsukuba” original dataset*


(a) Reference Frame **(b)** Match Frame

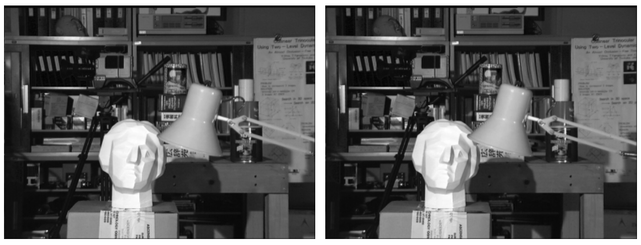
(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.3: *"tsukuba" dataset with CIE Y preprocessing*


(a) Reference Frame **(b)** Match Frame

(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.4: *"tsukuba" dataset with Gooch Color2Gray preprocessing*


(a) Reference Frame **(b)** Match Frame

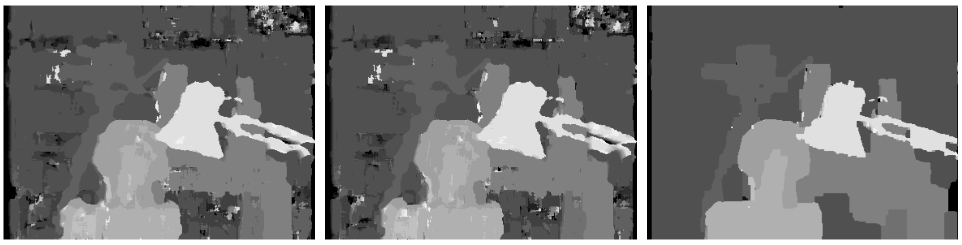
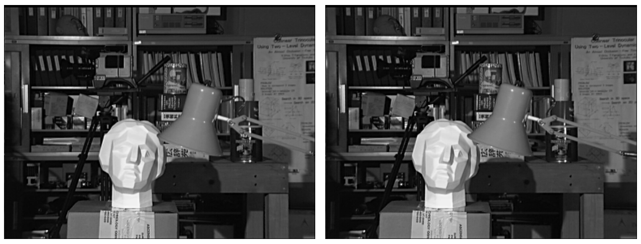

(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.5: *"tsukuba" dataset with Grundland Decolorize preprocessing*


(a) Reference Frame **(b)** Match Frame

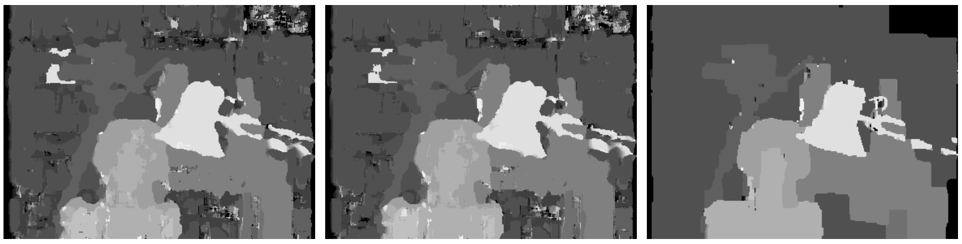
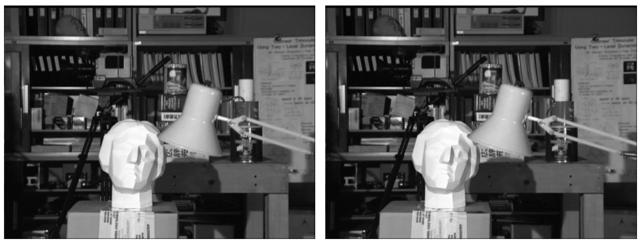

(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.6: *"tsukuba" dataset with Smith Apparent preprocessing*


(a) Reference Frame **(b)** Match Frame

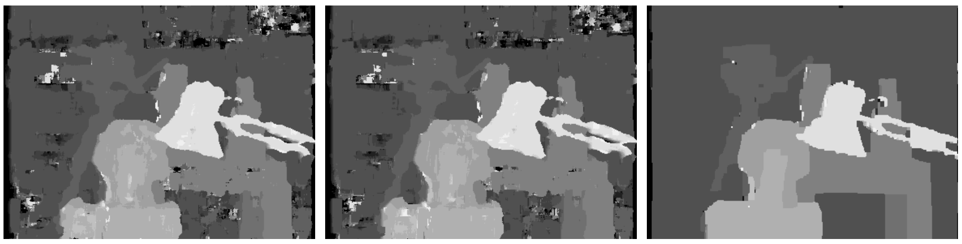
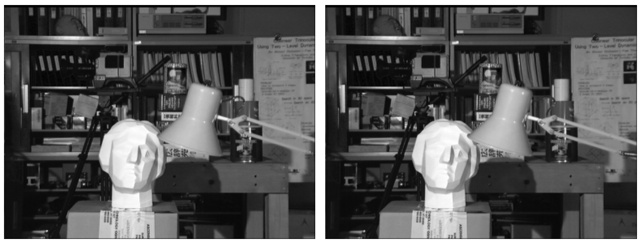

(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.7: *"tsukuba" dataset with Gray4Matching (linear) preprocessing*


(a) Reference Frame **(b)** Match Frame

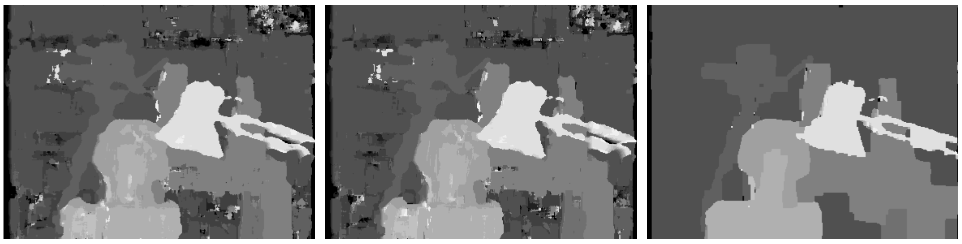

(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.8: *"tsukuba" dataset with Gray4Matching (sRGB) preprocessing*

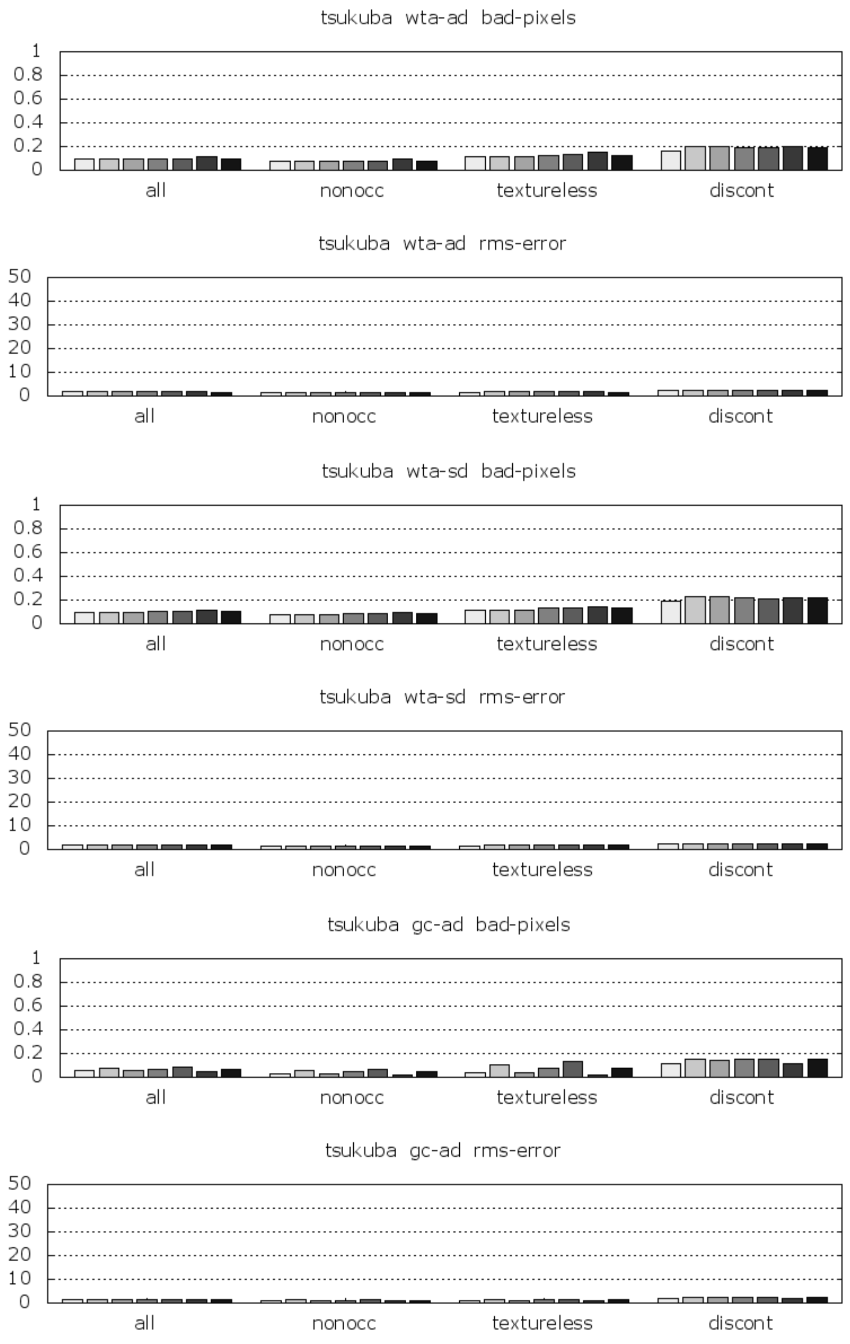


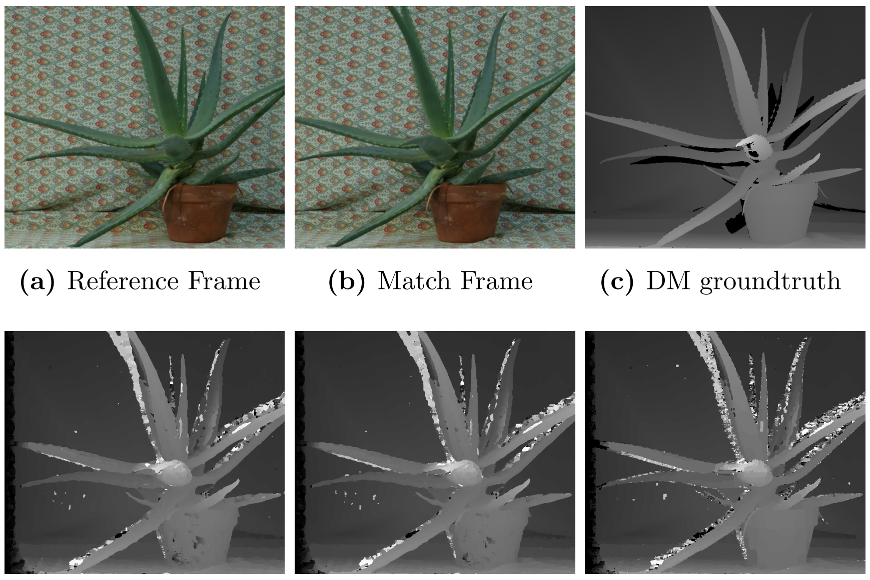
Figure 5.9: *Error measures for the "tsukuba" dataset. Legend is in Figure 5.1*

5.3.2 Results for dataset "aloe"

The "aloe" is the simplest of the 2006 datasets and the standard algorithms gives reasonable results with it.

The critical parts of this dataset are the discontinuities and the occluded areas on the border of the leaves, the grayscale versions have also problems with some isoluminant regions near a fold of the background cloth in the mid-low/right part of the image.

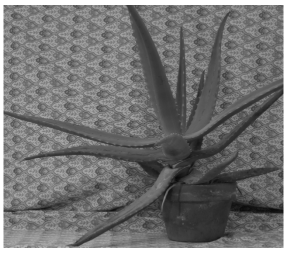
The best conversions seems to be Smith Apparent and the Gray4Matching version that assumes an sRGB colorspace, because, as previously said, the 2006 datasets are probably the only ones in sRGB.

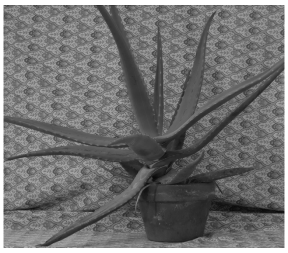


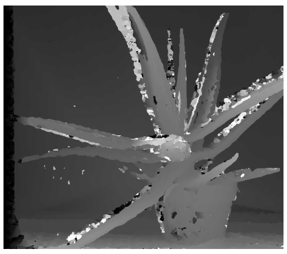
(a) Reference Frame **(b)** Match Frame **(c)** DM groundtruth

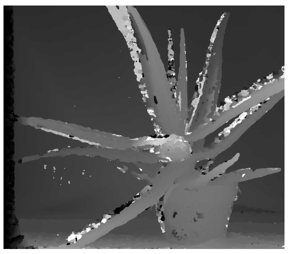
(d) DM from WTA-AD **(e)** DM from WTA-SD **(f)** DM from GC-AD

Figure 5.10: *"aloe" original dataset*


(a) Reference Frame


(b) Match Frame


(c) DM from WTA-AD


(d) DM from WTA-SD

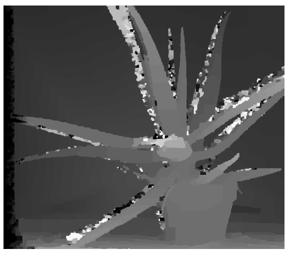
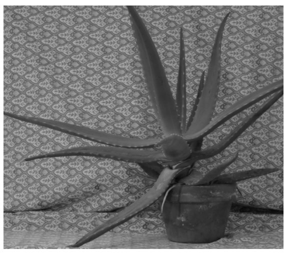
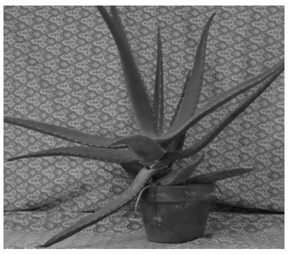
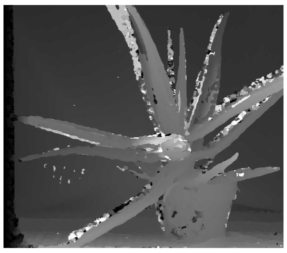
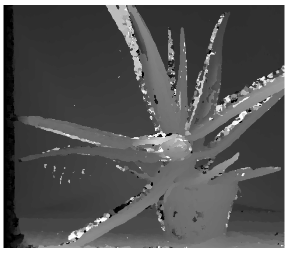

(e) DM from GC-AD

Figure 5.11: *"aloe" dataset with CIE Y preprocessing*


(a) Reference Frame


(b) Match Frame


(c) DM from WTA-AD


(d) DM from WTA-SD

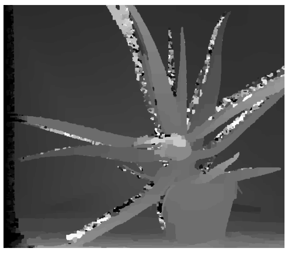
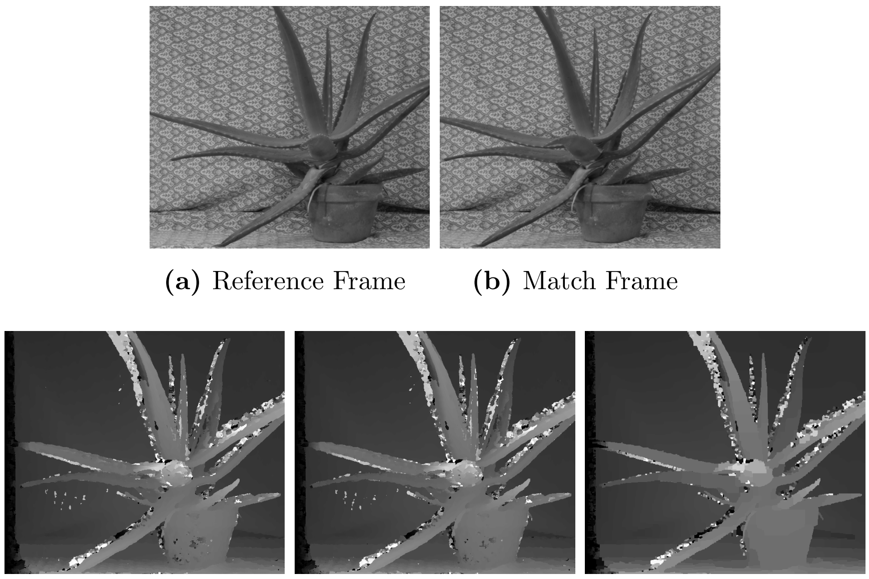
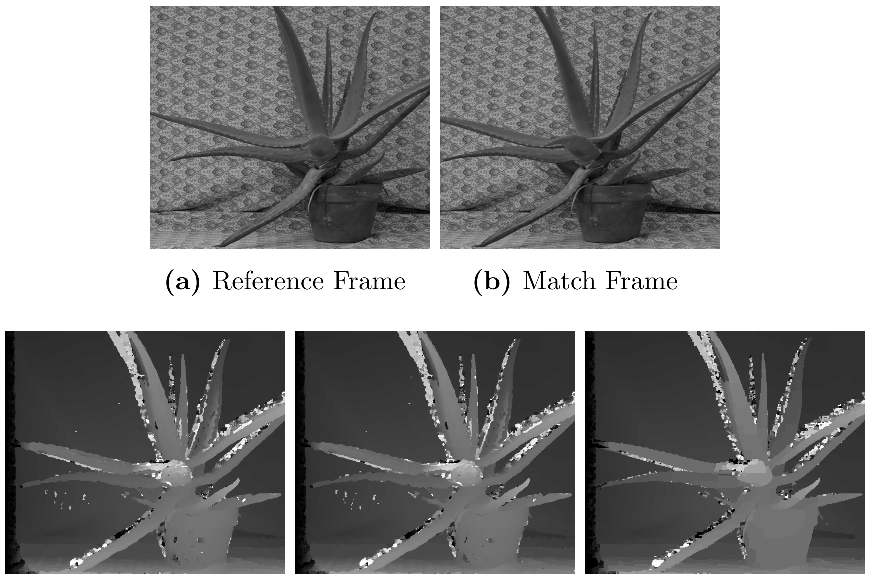

(e) DM from GC-AD

Figure 5.12: *"aloe" dataset with Gooch Color2Gray preprocessing*


(a) Reference Frame **(b)** Match Frame

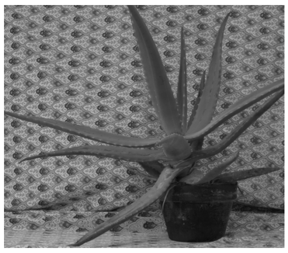
(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

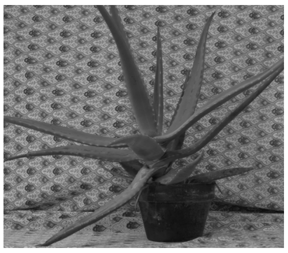
Figure 5.13: *"aloe" dataset with Grundland Decolorize preprocessing*

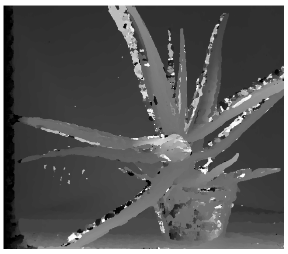

(a) Reference Frame **(b)** Match Frame

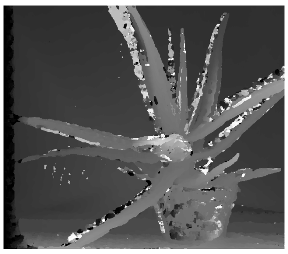
(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.14: *"aloe" dataset with Smith Apparent preprocessing*


(a) Reference Frame


(b) Match Frame


(c) DM from WTA-AD


(d) DM from WTA-SD

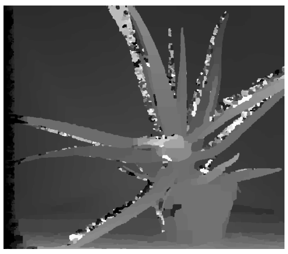
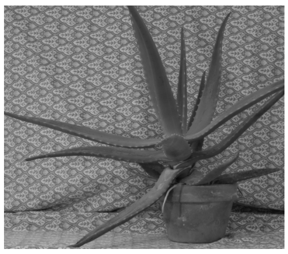
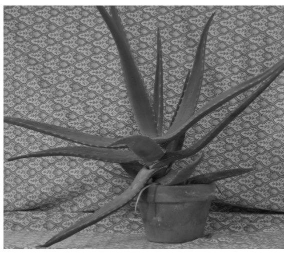
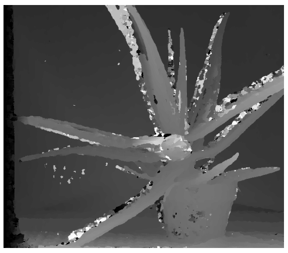
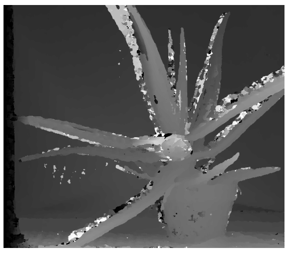

(e) DM from GC-AD

Figure 5.15: *"aloe" dataset with Gray4Matching (linear) preprocessing*


(a) Reference Frame


(b) Match Frame


(c) DM from WTA-AD


(d) DM from WTA-SD

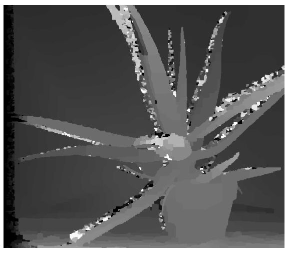

(e) DM from GC-AD

Figure 5.16: *"aloe" dataset with Gray4Matching (sRGB) preprocessing*

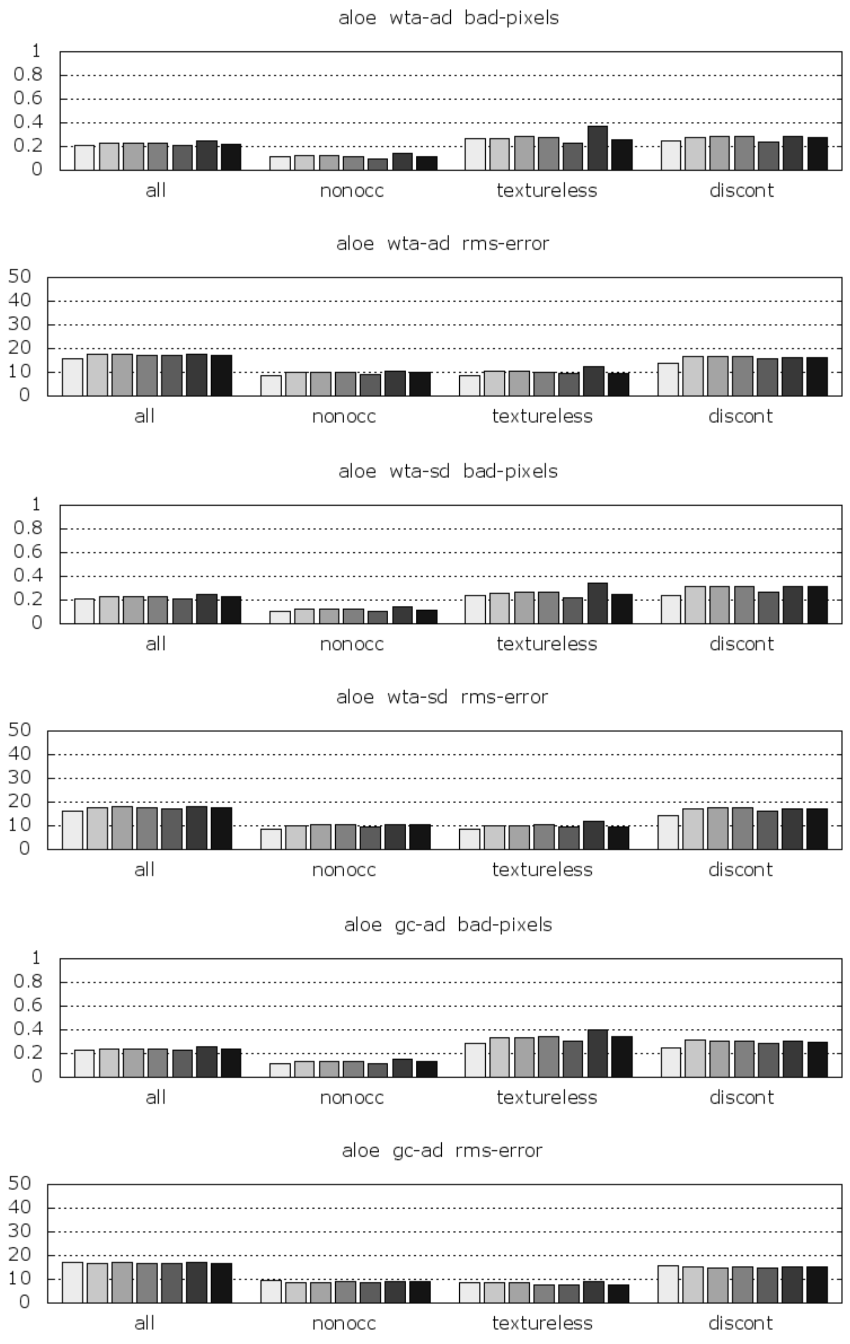


Figure 5.17: *Error measures for the "aloe" dataset. Legend is in Figure 5.1*

5.3.3 Results for dataset “dolls”

The “dolls” dataset, from the 2005 group, depicts a complex but still simple scenario, where the standard algorithms still gives reasonable results.

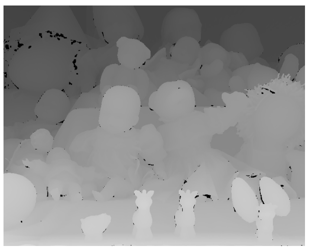
In this dataset, the benefits of Smith Apparent’s chrominance driven unsharp masking are evident even at a visual inspection, and their results are the best ones.

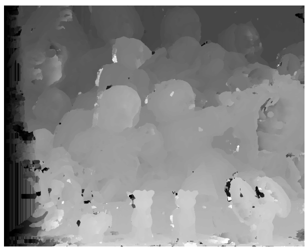
This is also one of the few datasets where the original color version obtains constantly the best results, helped by the “pastel” color gradients that predominates in the scene.

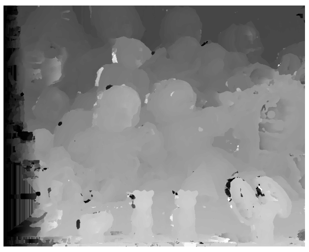
As the other 2005 datasets, “dolls” seems to have a linear gamma encoding.

(a) Reference Frame

(b) Match Frame


(c) DM groundtruth


(d) DM from WTA-AD


(e) DM from WTA-SD

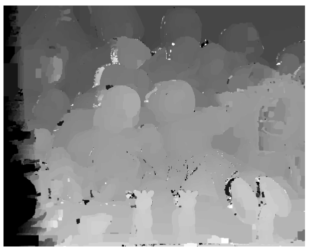
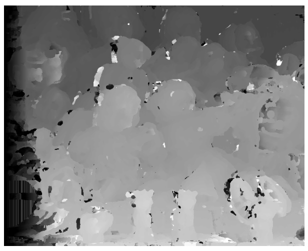

(f) DM from GC-AD

Figure 5.18: *“dolls” original dataset*

(a) Reference Frame

(b) Match Frame


(c) DM from WTA-AD

(d) DM from WTA-SD

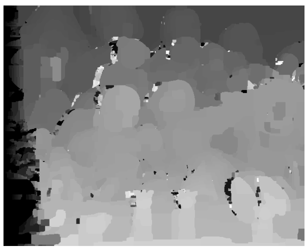
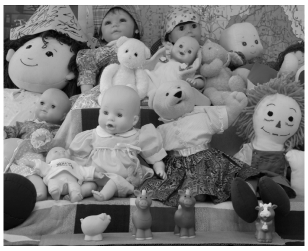

(e) DM from GC-AD

Figure 5.19: *"dolls" dataset with CIE Y preprocessing*


(a) Reference Frame

(b) Match Frame

(c) DM from WTA-AD

(d) DM from WTA-SD

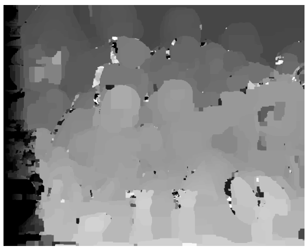
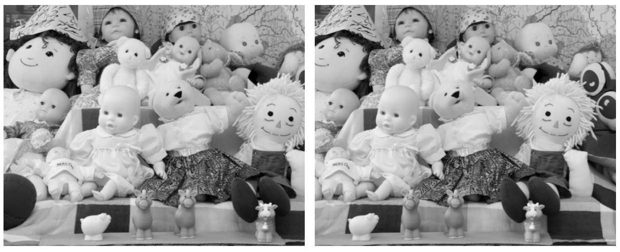

(e) DM from GC-AD

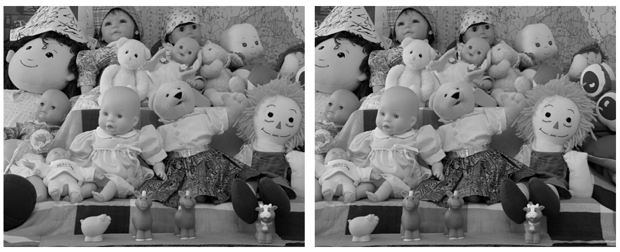
Figure 5.20: *"dolls" dataset with Gooch Color2Gray preprocessing*



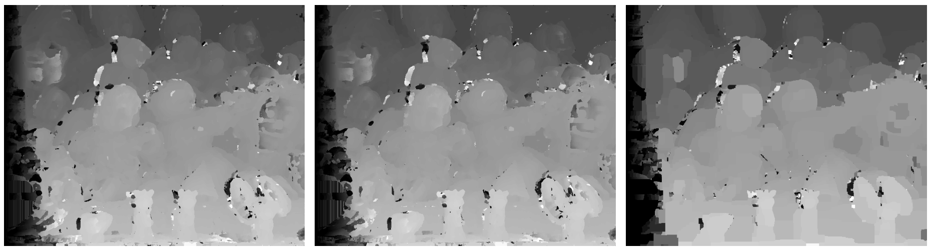
(a) Reference Frame **(b)** Match Frame

(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.21: *"dolls" dataset with Grundland Decolorize preprocessing*



(a) Reference Frame **(b)** Match Frame

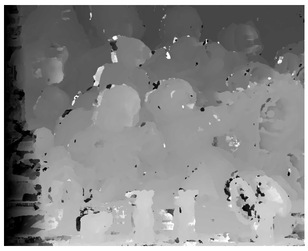


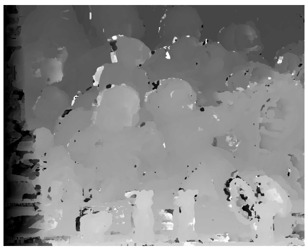
(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.22: *"dolls" dataset with Smith Apparent preprocessing*

(a) Reference Frame

(b) Match Frame


(c) DM from WTA-AD


(d) DM from WTA-SD

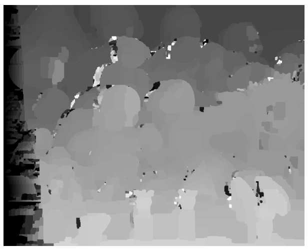
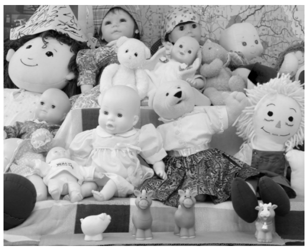

(e) DM from GC-AD

Figure 5.23: *"dolls" dataset with Gray4Matching (linear) preprocessing*


(a) Reference Frame

(b) Match Frame

(c) DM from WTA-AD

(d) DM from WTA-SD

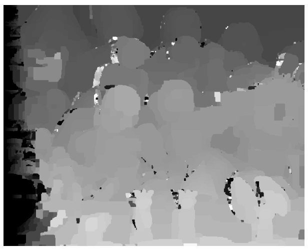

(e) DM from GC-AD

Figure 5.24: *"dolls" dataset with Gray4Matching (sRGB) preprocessing*

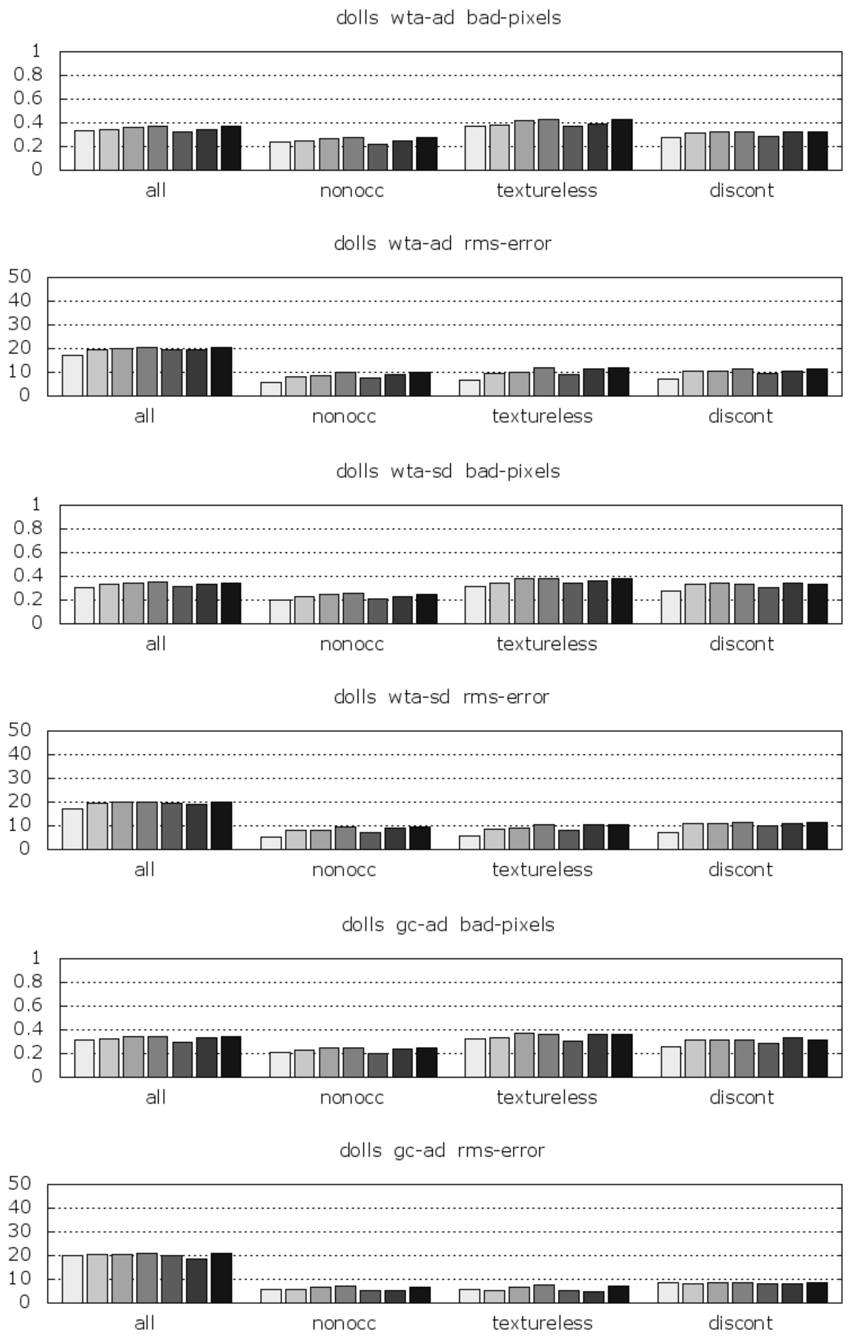


Figure 5.25: *Error measures for the "dolls" dataset. Legend is in Figure 5.1*

5.3.4 Results for dataset “laundry”

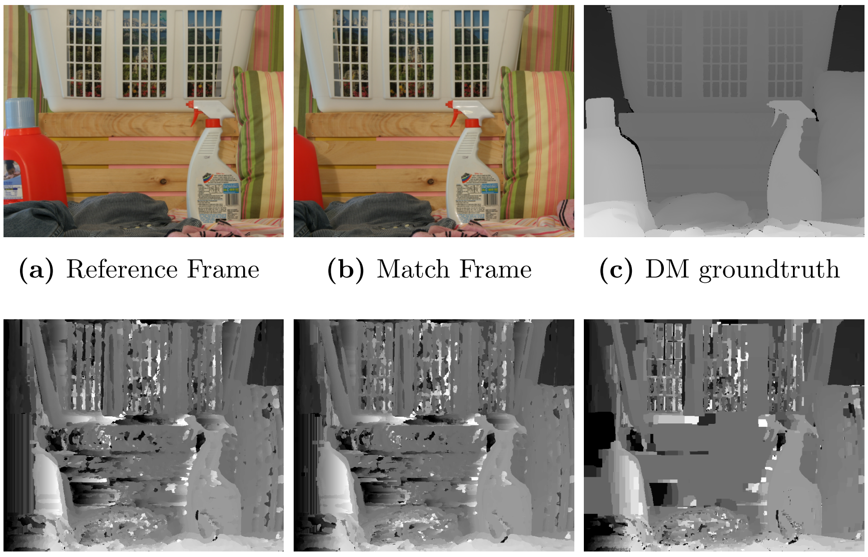
The “laundry” dataset, also from the 2005 group, contains elements, such as the background, that are really difficult for the the standard algorithms.

Our conversion is clearly the best one for this complex dataset, followed by Smith Apparent.

Another result that may seem surprising at a first examination is the relatively good performance of the CIE Y conversion with respect to Grundland Decolorize and Gooch Color2Gray.

The “disappearing” in the Match Frame of most of the red bottle at the left of the image is a typical cause of Grundland Decolorize’s weakness in the multi view applications, because the chromatic predominant axis of projection changes between the left and the right images.

As the other 2005 datasets, “laundry” seems to have a linear gamma encoding.



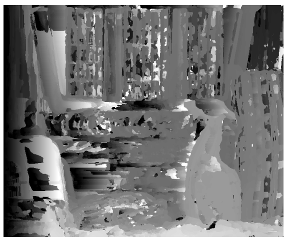
(a) Reference Frame **(b)** Match Frame **(c)** DM groundtruth

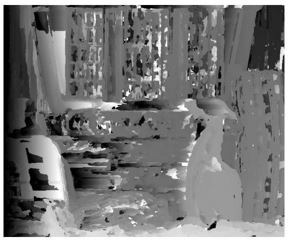
(d) DM from WTA-AD **(e)** DM from WTA-SD **(f)** DM from GC-AD

Figure 5.26: *“laundry” original dataset*

(a) Reference Frame

(b) Match Frame


(c) DM from WTA-AD


(d) DM from WTA-SD

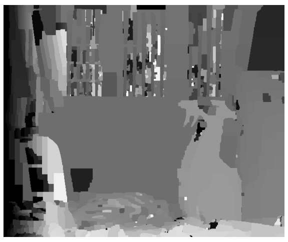
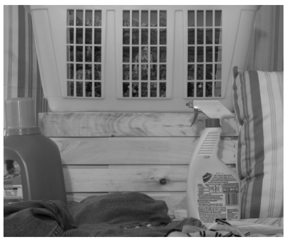
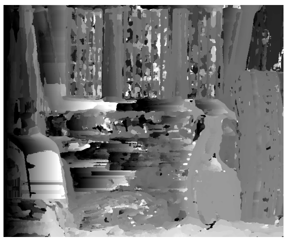
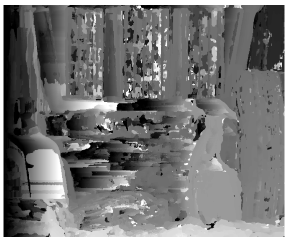

(e) DM from GC-AD

Figure 5.27: *"laundry" dataset with CIE Y preprocessing*


(a) Reference Frame

(b) Match Frame


(c) DM from WTA-AD


(d) DM from WTA-SD

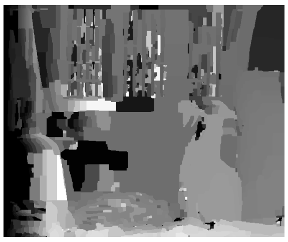
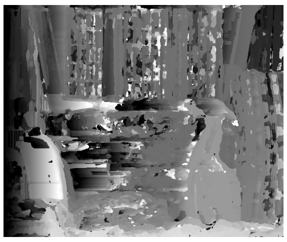
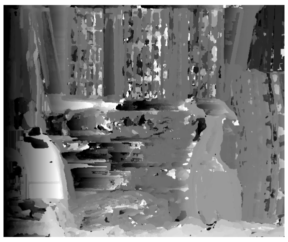

(e) DM from GC-AD

Figure 5.28: *"laundry" dataset with Gooch Color2Gray preprocessing*

(a) Reference Frame

(b) Match Frame


(c) DM from WTA-AD


(d) DM from WTA-SD

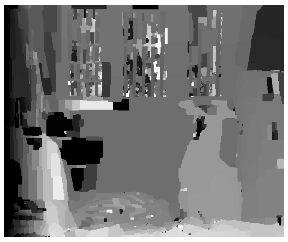
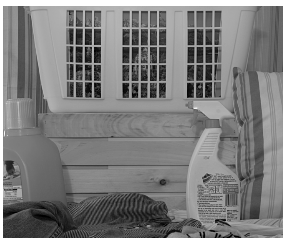
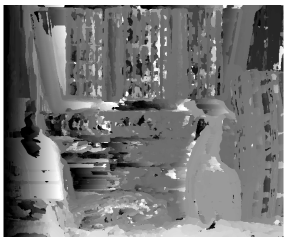
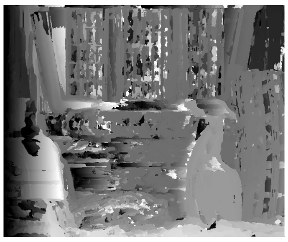

(e) DM from GC-AD

Figure 5.29: *"laundry" dataset with Grundland Decolorize preprocessing*


(a) Reference Frame

(b) Match Frame


(c) DM from WTA-AD


(d) DM from WTA-SD

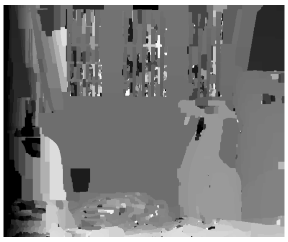
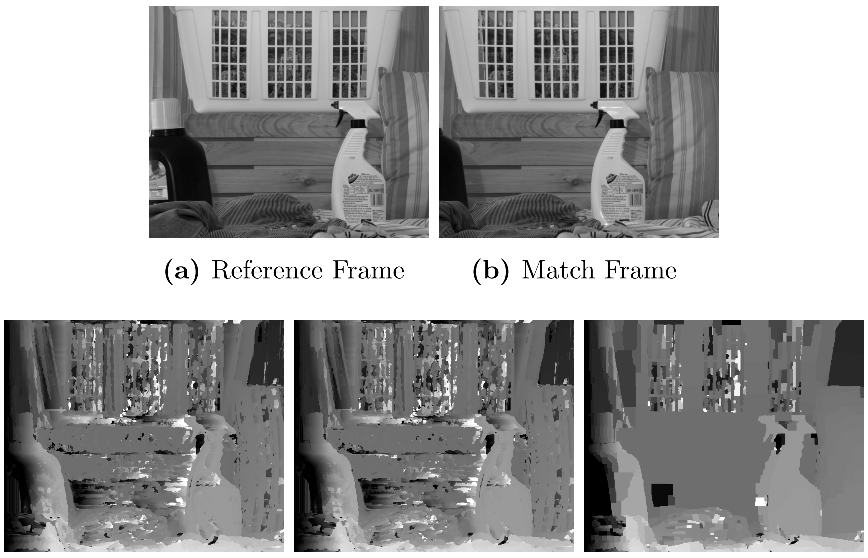

(e) DM from GC-AD

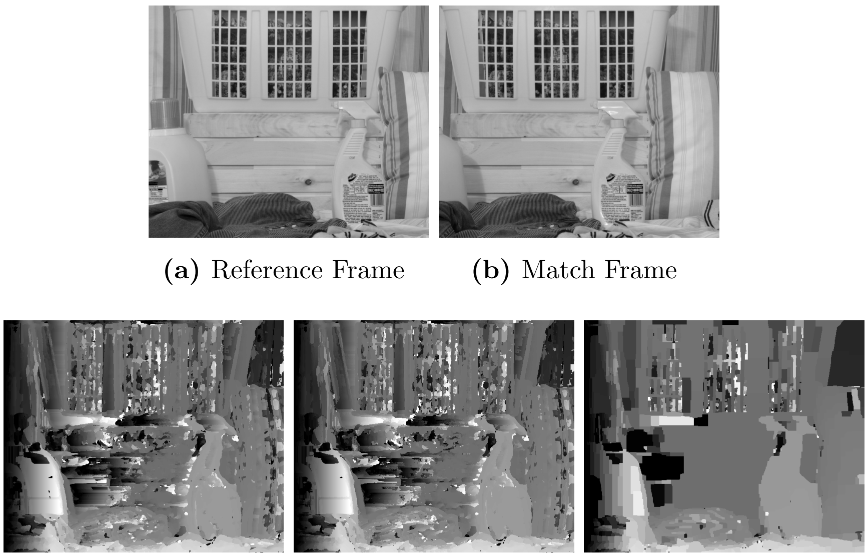
Figure 5.30: *"laundry" dataset with Smith Apparent preprocessing*



(a) Reference Frame **(b)** Match Frame

(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.31: *"laundry" dataset with Gray4Matching (linear) preprocessing*



(a) Reference Frame **(b)** Match Frame

(c) DM from WTA-AD **(d)** DM from WTA-SD **(e)** DM from GC-AD

Figure 5.32: *"laundry" dataset with Gray4Matching (sRGB) preprocessing*

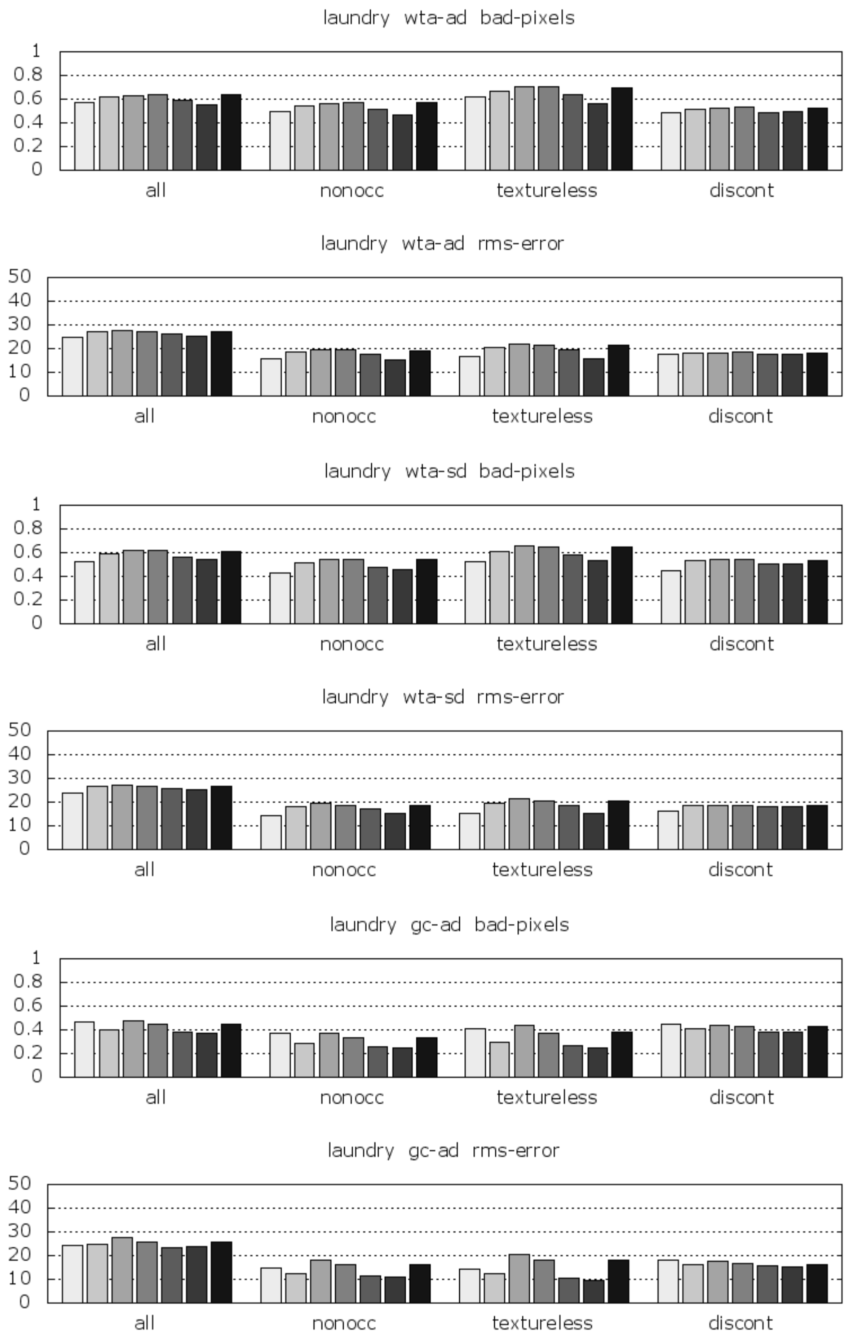


Figure 5.33: *Error measures for the "laundry" dataset. Legend is in Figure 5.1*

5.3.5 Results for the illumination variants

We show the partial results of an experiment in which the left and right images of the stereo pair to be matched are exposed to a heavy change of illumination.

While this situation rarely occurs in the stereo case, where just two images are included, when multiple images are used the illumination often changes.

How it can be clearly seen from the results, the standard StereoMatcher algorithms cannot cope with drastic changes, they are not robust to this kind of variations and the tested standard algorithms cannot obtain significant results.

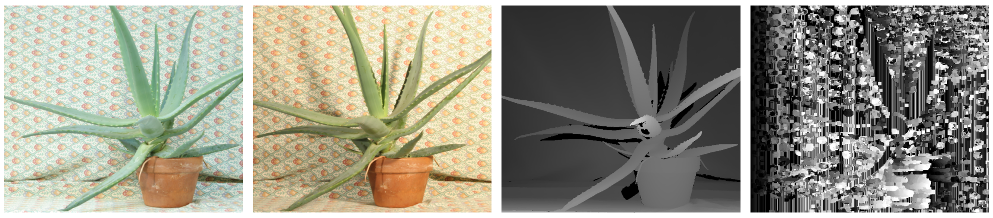
From our point of view we can see that every grayscale preprocessing helps, the worst results are the original color ones.

Future works are needed in order to show if, with more advanced (multi view) matching algorithms, our conversion helps.

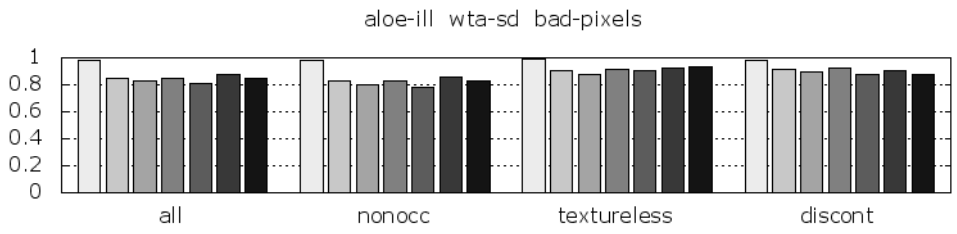
5.4 Matching results and perceptual results

In this section we discuss the similarities of our results with respect to an external study of the perceptual performances of many grayscale conversion that we used in this work. A quick description of the perceptual results will show how the matching results that we showed in this Chapter are surprisingly similar to the perceptual ones.

Grayscale conversion are often designed for the optimization of the “black and white” reproduction, for the preservation of overall perceptual properties, or with “artistic” purposes. Methods performing the conversion of color images to grayscale aim to retain as much information about the original color image as possible, while simultaneously producing perceptually plausible grayscale results.

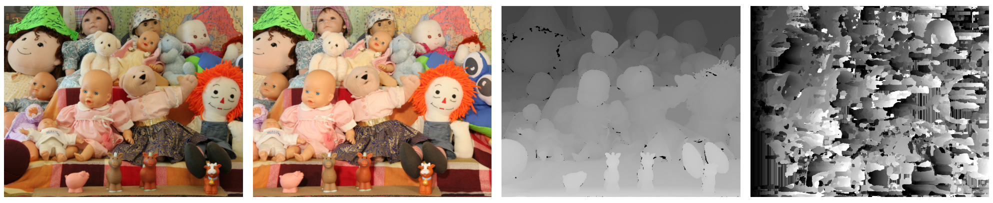


(a) Reference Frame **(b)** Match Frame **(c)** DM groundtruth **(d)** DM WTA-SD

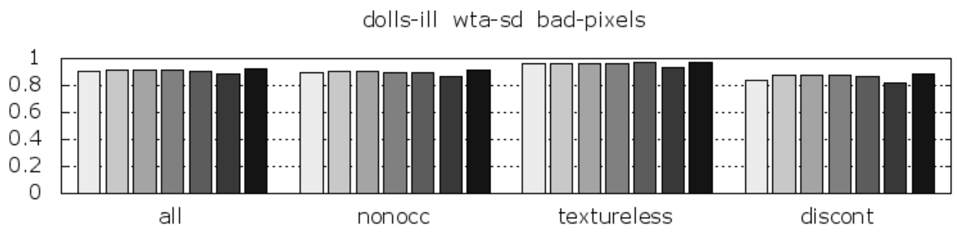


(e) Bad pixels for the WTA-SD algorithm

Figure 5.34: *Sample from the results of the illuminance-modified "aloe" dataset*



(a) Reference Frame **(b)** Match Frame **(c)** DM groundtruth **(d)** DM WTA-SD



(e) Bad pixels for the WTA-SD algorithm

Figure 5.35: *Sample from the results of the illuminance-modified "aloe" dataset*

In a recent paper [44], Čadík presents the results of two subjective experiments in which a total of 24 color images were converted to grayscale using seven grayscale conversion algorithms and evaluated by 119 human subjects using a paired comparison paradigm. Čadík surveyed nearly 20000 human responses and used them to evaluate the accuracy and preference of the color to grayscale conversions. To the best of our knowledge, the study presented in Čadík's paper is the first perceptual evaluation of color to grayscale conversions.

The grayscale conversion perceptually compared are CIE Y, Bala Spatial, Gooch Color2Gray, Rasche Monochromats, Grundland Decolorize, Neumann Adaptive and Smith Apparent. The results are summarized in Table 5.1 and in Figure 5.36[2].

It is really interesting that, in both the fields of application:

- Grundland Decolorize and our Gray4Matching variation of it are between the best conversions.
- Smith Apparent is one of the best conversion.
- Gooch Color2Gray gives poor results with respect to its own complexity.
- CIE Y performs well notwithstanding its simplicity.

These results

[2]In Čadík's image the conversions are called differently, but they are easily recognizable.

Grundland Decolorize	0.544	CIE Y	0.158
Smith Apparent	0.487	Gooch Color2Gray	0.149
Rasche05	-0.203	Bala Spatial	-0.819
Neumann07	-0.317		

Table 5.1: *Overall perceptual performances of the conversions inquired by Čadík. Results of the multiple comparison across all input images in both experiments. The conversions in the same boxes are considered perceptually similar.*

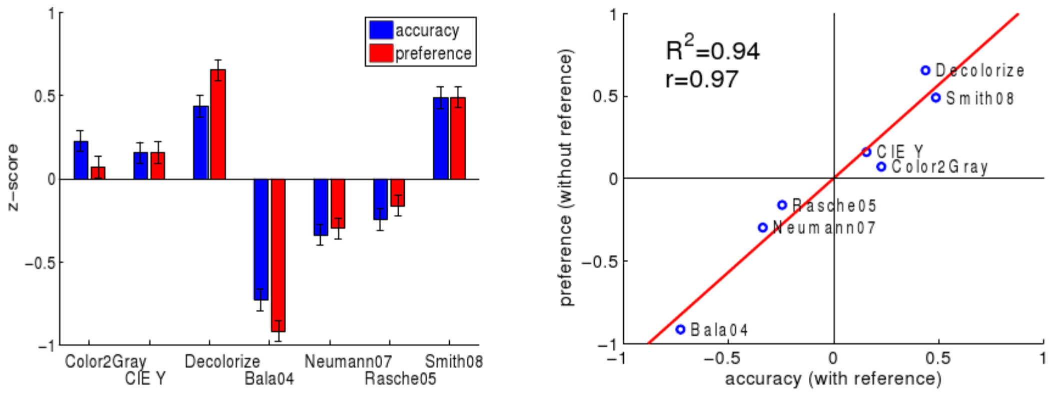


Figure 5.36: *The aggregated perceptual results taken from Čadík's paper. The original caption follows:* "Overall results separately for the two experiments. Left: overall scores for both the accuracy and preference experiments. Error bars show intervals of 95% confidence. Right: comparison of accuracy and preference experiments."

Chapter 6

Conclusions

According to our original objectives:

- We examined and implemented many different state of the art color to gray conversion algorithms.
- We designed and implemented Gray4Matching, a new grayscale method based on the best existing algorithm for our specific needs.
- We evaluated in a thorough way how the choice of different grayscale conversions affects the results computed by the standard stereo matching algorithms.

Our approach has been successful, the standard stereo matching algorithms that used our advanced grayscale conversion preprocessing often improved the results with respect to the original color images.

An unexpected and interesting side result is that the our stereo matching results coincides with the human perceptual preferences.

6.1 Future Work

This work laid the grounds for many possible future projects.

First of all, in the multi view stereo matching context of our long term PipeLab project[1] the benefits of our Gray4Matching consistent grayscale conversion will be much more evident even in standard scenarios. Thus, we plan to extend our evaluations to this much more complex and much more recent field, in order to find the grayscale conversion that is best suited for this difficult problem.

Our experimentation also highlighted the merits of Smith Apparent's chrominance driven unsharp masking, and this stimulated the design of a novel algorithm that will combine the advantages of this technique with the merits of our Gray4Matching conversion.

We also plan to reimplement more efficient versions of our algorithm in order to further improve its performance:

- A first improvement will allow a greater scalability in the number of the images, this will be accomplished by a more sophisticated *out of core* memory management that will use only the minimal needed amounts of memory.

- A second improvement will be the reimplementation of the algorithm in the *CUDA GPGPU* subset of the C language, in order to exploit the massive parallelization features of the recent Nvidia graphic cards.

Moreover, we are also planning the design of two more Gray4Matching approximated variations:

- A version that can process the images of the set *one by one* in by converting them in a *pipelined* way.

- A video conversion filter that can work in real time and can convert a video stream *keeping the temporal consistence* between the video frames.

[1]Briefly explained in Section 4.1.

6.2 Acknowledgments

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank the Visual Computing Laboratory of ISTI CNR for giving me the opportunity to work with them.

I am deeply indebted to my supervisors, Ing. Massimiliano Corsini and Dott. Paolo Cignoni, whose help, stimulating suggestions and encouragement helped me in all the time of research and writing of this thesis. Even if only Massimiliano and Paolo were my direct supervisors, *everyone* in the Laboratory helped me with their suggestions, their professionality (here my thanks goes especially to Marco Callieri, Fabio Ganovelli, Federico Ponchio and Matteo Delle Piane) and, more than any other thing, with their human, warm reception of me and the other thesis students between them.

I also want to thank my family, whose support has been more than essential. Especially, I would like to give my special thanks to Sara, whose patient love enabled me to complete this work.

Bibliography

[1] Roy S. Berns. *Billmeyer and Saltzman's Principles of Color Technology.* Wiley - Interscience, third edition, 2000.

[2] William David Wright. A re-determination of the trichromatic coefficients of the spectral colors. *Transactions of the Optical Society*, 30:141–164, 1928.

[3] John Guild. The colorimetric properties of the spectrum. *Philosophical Transactions of the Royal Society of London*, A230:149–187, 1931.

[4] Karl Rasche, Robert Geist, and James Westall. Detail preserving reproduction of color images for monochromats and dichromats. *IEEE Comput. Graph. Appl.*, 25(3):22–30, 2005.

[5] Raja Bala and Reiner Eschbach. Spatial color-to-grayscale transform preserving chrominance edge information. In *Color Imaging Conference*, pages 82–86, 2004.

[6] Amy A. Gooch, Sven C. Olsen, Jack Tumblin, and Bruce Gooch. Color2gray: salience-preserving color removal. *ACM Trans. Graph.*, 24(3):634–639, July 2005.

[7] Mark Grundland and Neil A. Dodgson. The decolorize algorithm for contrast enhancing, color to grayscale conversion. Technical Report UCAM-

CL-TR-649, University of Cambridge, Computer Laboratory, October 2005.

[8] Mark Grundland and Neil A. Dodgson. Decolorize: Fast, contrast enhancing, color to grayscale conversion. *Pattern Recogn.*, 40(11):2891–2896, 2007.

[9] Ricardo L. de Queiroz and Karen M. Braun. Color to gray and back: color embedding into textured gray images. *IEEE Transactions on Image Processing*, 15(6):1464–1470, 2006.

[10] Laszlo Neumann, Martin Čadík, and Antal Nemcsics. An efficient perception-based adaptive color to gray transformation. In *Proceedings of Computational Aesthetics 2007*, pages 73–80, Banff, Canada, 2007. Eurographics Association.

[11] Kaleigh Smith, Pierre-Edouard Landes, Joëlle Thollot, and Karol Myszkowski. Apparent greyscale: A simple and fast conversion to perceptually accurate images and video. *Computer Graphics Forum (Proceedings of Eurographics 2008)*, 27(2), apr 2008.

[12] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[13] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Found. Trends Comput. Graph. Vis.*, 3(3):177–280, 2008.

[14] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, 2002.

[15] Maarten Vergauwen and Luc Van Gool. Web-based 3d reconstruction service. *Mach. Vision Appl.*, 17(6):411–426, 2006.

[16] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press.

[17] Sylvie Chambon and Alain Crouzil. Color stereo matching using correlation measures. In *Complex Systems Intelligence and Modern Technological Applications - CSIMTA 2004, Cherbourg, France*, pages 520–525. LUSAC, sep 2004.

[18] M. Bleyer, S. Chambon, U. Poppe, and M. Gelautz. Evaluation of different methods for using colour information in global stereo matching approaches. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XXXVII, part B3a, pages 415–422, 2008.

[19] Trolltech®. QT® documentation.
`http://doc.trolltech.com/`.

[20] M.D. Fairchild and E. Pirrotta. Predicting the lightness of chromatic object colors using CIELAB. *Color Research & Application*, 16(6):385–393, 1991.

[21] Mark D. Fairchild. *Color Appearance Models*. Addison-Wesley, second edition, 2005.

[22] G. Wyszecki. Correlate for lightness in terms of CIE chromaticity coordinates and luminous reflectance. *Journal of the Optical Society of America*, 57(2):254–254, 1967.

[23] Y. Nayatani. Simple estimation methods for the Helmholtz-Kohlrausch effect. *Color Research & Application*, 22(6), 1997.

[24] Y. Nayatani. Relations between the two kinds of representation methods in the Helmholtz-Kohlrausch effect. *Color Research & Application*, 23(5), 1998.

[25] Rafał Mantiuk, Karol Myszkowski, and Hans-Peter Seidel. A perceptual framework for contrast processing of high dynamic range images. *ACM Transactions on Applied Perception*, 3(3):286–308, July 2006.

[26] Ali Alsam and Øyvind Kolås. Grey colour sharpening. In *Fourteenth Color Imaging Conference*, pages 263–267, Scottsdale, Arizona, Nov 2006.

[27] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, third edition, 2006.

[28] RD Nowak and RG Baraniuk. Adaptive weighted highpass filters using multiscale analysis. *IEEE Transactions on Image Processing*, 7(7):1068–1074, 1998.

[29] Bjarne Stroustrup. *The C++ Programming Language.* Addison-Wesley Professional, third edition, February 2000.

[30] The C++ Resources Network. cplusplus.com - The C++ Resources Network.
`http://www.cplusplus.com/`.

[31] Silicon Graphics Inc SGI. OpenGL.
`http://www.opengl.org/`.

[32] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh

processing tool. In *Sixth Eurographics Italian Chapter Conference*, pages 129–136, 2008.

[33] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. *Computer Science Tech. Report*, pages 94–125, August 1994.

[34] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[35] GEP Box and M.E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, pages 610–611, 1958.

[36] MJ Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–91, 1996.

[37] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *International Journal of Computer Vision*, 28(2):155–174, 1998.

[38] L. Matthies, T. Kanade, and R. Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–238, 1989.

[39] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision*, 35(3):269–293, 1999.

[40] Yuichi Nakamura, Tomohiko Matsuura, Kiyohide Satoh, and Yuichi Ohta. Occlusion detectable stereo – occlusion patterns in camera matrix. In *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, pages 371–378, Washington, DC, USA, 1996. IEEE Computer Society.

[41] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:195–202, 2003.

[42] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[43] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[44] Martin Čadík. Perceptual evaluation of color-to-grayscale image conversions. *Comput. Graph. Forum*, 27(7):1745–1754, 2008.

[45] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions via graph cuts. Technical report, Ithaca, NY, USA, 2001.