High Dynamic Range Panoramic Imaging with Scene Motion

prepared by

Simon Silk

supervised by

Jochen Lang

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Abstract

Real-world radiance values can range over eight orders of magnitude from starlight to direct sunlight but few digital cameras capture more than three orders in a single Low Dynamic Range (LDR) image. We approach this problem using established High Dynamic Range (HDR) techniques in which multiple images are captured with different exposure times so that all portions of the scene are correctly exposed at least once. These images are then combined to create an HDR image capturing the full range of the scene. HDR capture introduces new challenges; movement in the scene creates faded copies of moving objects, referred to as ghosts.

Many techniques have been introduced to handle ghosting, but typically they either address specific types of ghosting, or are computationally very expensive. We address ghosting by first detecting moving objects, then reducing their contribution to the final composite on a frame-by-frame basis. The detection of motion is addressed by performing change detection on exposure-normalized images. Additional special cases are developed based on a priori knowledge of the changing exposures; for example, if exposure is increasing every shot, then any decrease in intensity in the LDR images is a strong indicator of motion. Recent Superpixel over-segmentation techniques are used to refine the detection. We also propose a novel solution for areas that see motion throughout the capture, such as foliage blowing in the wind. Such areas are detected as always moving, and are replaced with information from a single input image, and the replacement of corrupted regions can be tailored to the scenario.

We present our approach in the context of a panoramic tele-presence system. Tele-presence systems allow a user to experience a remote environment, aiming to create a realistic sense of "being there" and such a system should therefore provide a high quality visual rendition of the environment. Furthermore, panoramas, by virtue of capturing a greater proportion of a real-world scene, are often exposed to a greater dynamic range than standard photographs. Both facets of this system therefore stand to benefit from HDR imaging techniques.

We demonstrate the success of our approach on multiple challenging ghosting scenarios, and compare our results with state-of-the-art methods previously proposed. We also demonstrate computational savings over these methods.

Acknowledgements

First and foremost, I would like to express my utmost gratitude to my supervisor, Dr. Jochen Lang, who has been extremely generous with his time and expertise and has provided invaluable feedback and guidance to help me past difficult points in my research and writing.

Second, I would like to thank Xida Chen (University of Calgary) and Miguel Granados (Max Planck Institut) for making the code for their respective background estimation approaches available so that I could compare my results to theirs. I would also like to thank Henning Zimmer (Saarland University) for taking the time to run several of my image sets through his HDR alignment code for further comparison. The generosity of these fellow researchers has allowed me to write a much stronger thesis.

Finally, I thank my family, immediate and extended, for their ongoing support throughout my education.

Contents

List of Figures

List of Notations

Δt	Exposure time
E	Irradiance, at the image sensor
$f(\cdot)$	Camera's *forward* response function; maps irradiance to digital pixel values
$g(\cdot)$	Camera's *reverse* response function, mapping digital pixel values to irradiance, and referred to as response function: $g(\cdot) = f^{-1}(\cdot)$
H	Floating point (HDR) pixel value
i, j	Horizontal and vertical pixel indices, respectively. When pixels have no spatial inter-dependence, j may be omitted, in which case i is a linear index of all pixels, wrapping at the end of rows
k	Colour channel index
$n \in [1, N]$	Index for N exposure-bracketed images. When used as a single subscript on a matrix, indicates the entire corresponding input image $\mathbf{Z}$ or normalized image $\mathbf{E}$.
X	Exposure, $X = E\Delta t$
z	Integer digital pixel value, $Z_{min} \leq z \leq Z_{Max}$

Chapter 1

Introduction

High Dynamic Range Imaging (HDRI) is the computational photography technique of capturing a series of images at different exposures, e.g. at different shutter speeds, so that all parts of the scene are correctly exposed in one or more images. These images can then be combined in post-capture processing to form an image with an expanded dynamic range, where dynamic range is the ratio of intensity of the brightest to darkest points in the image. As early as 1857, photographer Gustave LeGray manually combined multiple differently exposed images of the same scene to extended the effective dynamic range of film [1]. The modern foundations of HDRI were published in 1997 [2], providing a method to recover unknown digital camera or film response curves, and to use those recovered curves to normalize differently-exposed images of a scene before taking their weighted average to produce a final HDR image. Various applications quickly sprang up including the use of HDR images of real scenes for image-based relighting of 3D models, artistic photography of scenes in which the range of radiance values exceeds the dynamic range of the camera, and many scientific applications such as astronomy and medical imaging. In particular, the film industry was an early adapter of HDR imaging as evidenced by the fact that the popular HDR image storage format, OpenEXR, was developed and released by special effects company Industrial Light and Magic [3]. HDR imaging is currently used in many image-based modeling approaches in computer graphics.

Regarding image-based modeling, recent years have seen growing interest in panoramic tele-presence systems. Tele-presence systems allow a user to experience an environment remotely, "being there" from a distance. Familiar examples are Google Street View and Microsoft Bing Streetside. Tele-presence is a relatively new field that has already shown itself to have multiple applications. Examples include recreation, allowing people to ex-

plore far away locations without needing to be physically present; business use, allowing employees in different geographic regions to meet more regularly; and accessibility, allowing the public to visit sites of artistic or historic interest which might otherwise be damaged by regular human presence.

A true sense of "being there" requires high fidelity reproduction of the stimulus for all senses; a major component of this is visual fidelity, and a key component of visual fidelity is the capture of the full range of intensities available in the scene. While current cameras cannot capture this complete range, a multi-exposure HDRI process can. The integration of this approach into existing tele-presence projects is therefore a natural extension that can greatly enhance the sense of "being there." Furthermore, panoramas capture a greater proportion of the scene and therefore generally experience a greater dynamic range than standard photographs, further necessitating HDR techniques.

In this thesis, we outline a High Dynamic Range Imaging capture and processing pipeline, with an eye to applying it to panoramic tele-presence, primarily for capture of public spaces such as outdoor environments or inside publicly acessible buildings. We show that HDRI can increase the image quality of panoramas captured for tele-presence systems, particularly in the case where the hardware being used may have serious limitations.

1.1 Problem Definition

Our goal is to capture panoramic images with high dynamic range using a Ladybug2 camera by Point Grey Research[1]. The Ladybug2 spherical camera, shown in Figure 1.1, comprises six cameras in a single compact housing. Five cameras form a horizontal ring looking outward, while a sixth looks upward; together, they capture about 75% of a complete sphere. However, the imaging sensors only provide 8-bit data [4]. Given the linear response of CCDs [5, Ch.5], this means a maximum of 255:1 contrast even assuming no noise, where we calculate dynamic range as the ratio of highest value to lowest non-zero value. Considering that many digital cameras provide 12-16 bits of RAW data and still benefit from HDR methods, the Ladybug2 clearly stands to benefit from HDRI.

While the naive approach to HDRI, in which we assume constant illumination and no movement of camera or scene elements, has been well explored [2, 6, 7], problems

[1] http://www.ptgrey.com/

arise as soon as any of these assumptions is broken. For our work, we assume constant global illumination since this is a key assumption allowing recovery of camera response curves and normalization of the images in exposure space. We also assume a static camera. However, we do not assume a static scene and therefore extra processing must be performed to reduce or eliminate the repetition or blurring of objects that are moving in the scene during capture. This process has come to be known as "ghost removal" or "deghosting" since the repeated copies of moving objects are made semi-transparent and ghostly looking by the HDR weighted averaging process. Our primary work and contributions are in ghost removal.

Figure 1.1: Ladybug2 spherical camera

1.2 Background

Tele-presence benefits greatly from the integration of HDR imaging techniques. Since panoramic tele-presence systems often involve capture in public spaces, ghosting is a frequent, if not constant, problem, and deghosting methods are therefore a requirement. Furthermore, since it may also be desirable to do augmented reality (addition of virtual objects to the real images), or to use the captured HDR panoramas for other applications such as image-based relighting, it would be highly advantageous to go a step further and perform background modeling. This section provides a brief introduction to the related literature, which is expanded upon in Chapter 2.

Numerous single-shot HDR sensors have been proposed in the literature [8–19], but most are not yet a commercial reality, and the few that are are of low resolution and are primarily aimed at specialized applications in the automotive and security industries. A few cameras have also been released offering HDR or HDR-like image capture [20–22], notably the Point Grey Research Ladybug2 used in our work.

Previous work has explored the use of the Ladybug2 in tele-presence [23–25]. The RAW images from the camera's 6 CCDs are captured along with exposure data, and GPS and inertial sensor readings from additional devices. The camera's SDK provides functionality for colour processing and remapping of CCD views into standard panoramic projections such as cubic or equirectangular. The combination of colour-processed panoramas and geographic meta data are then presented to a user through a GUI for the purposes of exploring distant environments, i.e., tele-presence.

Methods for recovery of camera response curves from differently-exposed images, and the use of these curves to combine these images into an HDR image, are well explored in the existing literature [2, 6, 7, 26]. Generally, the input images are assumed to sample an unknown response function at various points; these samples are used to fit a general curve model. The fitted curve, called the response curve, can then be used to convert digital pixel values, whose response to light incident on the sensor may be non-linear, into values linearly related to incident light. These values are in turn normalized through division by exposure time and/or gain. Finally, for each pixel, a weighted average of the pixel's values across the input images is taken to produce a floating point value. The weighting function is chosen so as to reduce the contribution of values captured by unreliable portions of the response curve. We will refer to this basic process as the naive HDR approach. Details of how to handle degenerate cases, such as a pixel having all-zero weights, are not covered in the existing literature, so we address these in our work.

When the input images are captured in a real-world environment, there is often motion in the scene, e.g. due to people, animals, and vehicles moving in the scene. The result after weighted averaging is faded copies of rapidly moving objects, or blurring of slower moving or stationary but deforming objects; these artifacts are known as ghosts. Ghost removal approaches primarily break down into optical flow-based approaches [27–29], and change detection-based approaches [30–32]. In the first category, optical flow algorithms are used to realign pixels corresponding to moving objects before applying the naive HDR approach. In the latter category, some variation on change detection is applied to find areas that have seen moving objects and these areas are then replaced with information from one or more input images so as to either remove the ghosts or at least fix

them in one place and reduce repetitions. More state of the art techniques include graph cut approaches to background estimation, modified for application to HDRI [33,34], and optical flow methods using recent energy-based algorithms [29].

1.3 Thesis Statement

HDR imaging enhances realism in panoramic tele-presence applications, but capturing large data sets in public spaces necessitates an efficient and high quality ghost removal algorithm. Change detection-based deghosting can meet this challenge, delivering state-of-the-art results with a computational advantage over graph cuts-based deghosting, and with a careful analysis of the image acquisition process it can handle various challenging scenarios.

1.4 Thesis Overview

Figure 1.2 shows a high level overview of the proposed system. Following a standard image-based rendering pipeline, there is a capture stage, a processing stage, and a display stage. The capture stage produces exposure-bracketed RAW (Bayer-mosaiced) images, with associated exposure information, specifically exposure time and gain.

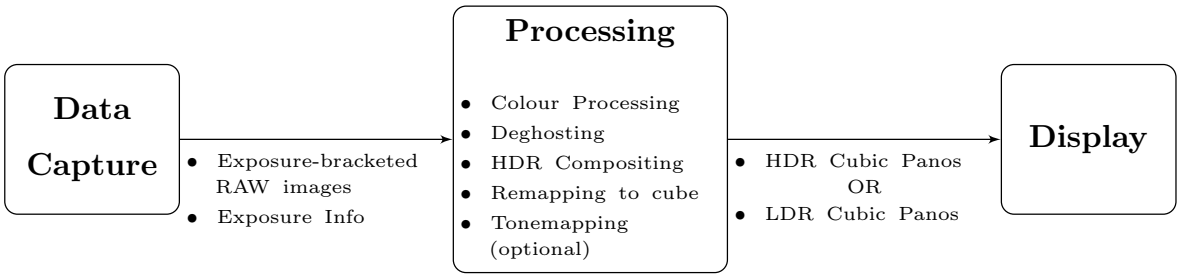


Figure 1.2: High level system overview

The majority of the proposed system falls within the processing block, which comprises several stages. The first key stage is colour processing in which we convert from RAW sensor images to RGB colour images. A second and critical stage is de-ghosting in which we correct artifacts introduced by objects moving in the scene during the exposure-bracketed image capture. HDR compositing is the stage in which the images are converted into linear radiance values, normalized in exposure space, and averaged together

to create a final image with greater dynamic range. Finally, the views of the six camera sensors are re-projected onto the six faces of a standard cubic panorama and the HDR panorama is saved to disk.

The purpose of the display stage is to take the cubic panorama saved to disk and display it on screen. Since even modern LCD displays are typically limited to a dynamic range of about two orders of magnitude [30, Ch.1], they cannot directly display an HDR image which means that at some point between HDR compositing and display, the HDR image's value must be compressed back into a displayable range (typically 8 bits), a process known as tonemapping. Therefore, an optional stage before display is to tonemap the images before saving them as a cube if they are to be viewed in a standard LDR panorama viewer. Ortherwise, the HDR panoramas can be input direclty to an HDR panorama viewer capable of real-time tonemapping.

In Chapter 2, we provide detailed reviews of the existing ghost removal techniques. Regarding HDR background estimation, we find that this is done quite well by state of the art graph cut approaches [33], but these are still too computationally expensive for processing of large data sets ranging into the thousands of images, and even with these powerful approaches, ghosting artifacts can still remain.

We therefore revisit the concept of change detection-based deghosting, which has the advantage of being relatively fast. We improve upon existing methods through a careful consideration of what can be known about pixel values in images with changing exposure, as well as through the integration of recent fast "superpixel" segmentation techniques. We show that this combination can provide good "best-effort" background estimation results in most scenarios, modeling the background in most areas, and in areas where that is difficult or impossible, freezing ghosts to reduce replication.

We categorize ghosts as displaying either "discrete motion" or "fluid motion." Discrete ghosts are those that primarily displace large amounts between frames, such as people walking or cars driving. Fluid ghosts are those that primarily stay in one place but deform, such as waves on water, foliage blowing in the wind, and people standing still but gesticulating. We show that discrete ghosts can be removed on a frame-by-frame basis so as to model the background in those regions without severely impacting dynamic range. We show that fluid ghosts can be detected using simple methods and frozen so as to reduce blurring even though this means we do not always strictly model the background where fluid motion has occured, hence the term "best-effort" background estimation.

1.5 Contributions

The following outlines the contributions of our work:

- Change detection in exposure-bracketed sequences based on a deliberate logical analysis of how pixel values should behave within an exposure-bracketed image stack and what can be known about pixels with respect to under-/over-exposure, and known exposure times.
- Use of recent Superpixel segmentation techniques to refine change detection results.
- Classification of ghosts as either "discrete" (ghosts that displace by large amounts between each frame) or "fluid" (ghosts that stay in one place but visually deform or fluctuate, such as foliage or flags blowing in the wind).
- A fast novel technique for detecting fluid areas and choosing a single input image to replace them so that ghosts in those regions are frozen.

1.6 Thesis Organization

The rest of this thesis is organized as follows.

- **Chapter 2** reviews in detail the related work introduced above, providing a thorough background on HDR imaging and ghost removal, and a brief overview of over-segmentation and tone mapping.
- **Chapter 3** outlines our capture hardware setup and associated capture code, as well as naive HDR combination and panorama creation for our system.
- **Chapter 4** introduces the proposed background estimation technique for Low Dynamic Range (LDR) images.
- **Chapter 5** extends the LDR approach to handle ghost removal for HDR imaging and shows initial results.
- **Chapter 6** provides more results, compares the proposed method to recent approaches from the literature as well results of commercial HDR software, and discusses limitations of the proposed method.

To improve readability, a list of notations is provided on page ix; as much as possible, our literature review and our own work will adhere to this notation, with deviations being noted in the text. When presenting HDR results, they must be tonemapped for viewing on screen or in print. Unless otherwise noted, all results are tonemapped in HDRsoft's[2] *Photomatix Pro* using the *Details Enhancer* tonemapper with either default or *Smooth Skies* settings.

[2] http://www.hdrsoft.com/

Chapter 2

Related Work

Our work has focused on the High Dynamic Range (HDR) aspects of HDR tele-presence, so we provide a thorough overview of HDR literature, including proposed approaches for one-shot HDR capture, fundamentals of naive HDR combination in which a static camera and scene are assumed, and approaches to ghost removal techniques which attempt to compensate for scene elements moving during the capture of exposure-bracketed images. Naive HDR combination in the context of our system is detailed in Section 3.4.

2.1 One-Shot HDR Capture

Here, we provide a review of several alternatives to the exposure-bracketed HDR capture process. We use One-Shot HDR to refer to devices that allow the user to effectively capture an HDR image through a single capture event. One option in photographic HDR literature is to create HDR images directly from RAW images. Various HDR sensors have been proposed in the existing research, and several specialized cameras exist that can capture HDR or at least extended dynamic ranges directly. In addition, consumer grade cameras have begun to appear with so-called HDR modes. Finally, the Ladybug2 offers two HDR modes, which will be detailed in Section 3.1.

2.1.1 HDR from a Single RAW Image

In the community of artistic photographers using HDR imaging, there is a common misconception that an HDR image can be created from a single RAW image, removing the need for multiply-exposed JPEG images [35, Ch.5]. The logic is that the JPEG images produced by cameras are typically 8-bit, affording a dynamic range of about

250:1, while many cameras have RAW-image bit-depths of 10-14 bits, affording dynamic ranges of 1,000-16,000:1.

The misconception is two-fold. First, there is the incorrect assumption that the JPEG images' pixel values are linearly encoded, when in fact they typically represent a non-linear remapping of the RAW values so much of the range is still represented. The mapping generally includes a gamma function as well as other unknown image processing functions performed on-camera. While the mapping may sacrifice details in some portions of the RAW image range, the resulting JPEG typically captures most of the range of the RAW image.

Second, there is the assumption that the entire range of a sensor's output represents valid data so that, for example, if we have a 10-bit sensor, the dynamic range we can expect from a single RAW image is $(2^{10} - 1) : 1 = 1,023 : 1$, where the dynamic range is calculated using the second-lowest value to avoid the trivial result of infinite range. In fact, imaging systems always have a constant noise floor which greatly reduces the dynamic range of the sensors. In this example, if the noise floor has a mean value of 3, this should be assumed to reduce the usable dynamic range of the sensor by at least 3-fold to about $340 : 1$ or worse.

So, while some dynamic range improvement is possible using the RAW image, an HDR image created from a single RAW image and showing greater dynamic range than the corresponding JPEG likely improved as much due to the more powerful processing tools available on a PC as it did from the higher bit-depth of the RAW image. Additionally, any dynamic range gains possible through this process will generally boost the image into a medium dynamic range rather than high dynamic range.

2.1.2 HDR Sensors

While a true HDR sensor would have pixels that directly capture a high dynamic range, no such sensors appear to exist at present. However, various proposed systems can capture HDR data using specialized pixel configurations or readers, or modifications external to the sensor.

Mitsunaga and Nayar [8] propose spatially varying pixel exposures, in which a mask with a known spatially varying pattern of optical transmittance is placed over the CCD or CMOS, attenuating the light hitting some pixels more than others. An HDR image can be reconstructed by demosaicking the intensities, similar to how the grayscale output of a Bayer-tiled CCD is demosaicked to produce colour information. For an experimental

8-bit sensor with four attenuation tiles increased in steps of 4x (2EV), they show that a dynamic range of over 800:1 can be expected, as compared to the 255:1 maximum of the 8-bit sensor alone. Nayar and Branzoi later improved this method with a mask whose transmittance can be modified dynamically at capture time [9].

Another approach is to use beam splitters to split the incoming image into multiple copies which can each be focused on a different CCD with different exposure settings [10, 11]. This allows one-shot capture, but is expensive as it requires precise alignment of the various optical elements, and likely would require at least three sensors to achieve a significant improvement in dynamic range. Methods have been proposed wherein each pixel is composed of multiple sensor wells, typically two, each with a different size and hence sensitivity. The values read by the two can be combined to achieve HDR or at least medium dynamic range information [12]. This style of sensor made a commercial appearance in the Fujifilm Super CCD and EXR sensors. Another approach has been proposed in which each pixel has a timer that counts the time required to saturate, which is proportional to the irradiance at that pixel [13]. This is effectively the same as each pixel having independent exposure control.

Yet another method to improve dynamic range is by direct conversion of photocurrent to voltage, taking advantage of the logarithmic $I - V$ relationship in MOS sensors to perform logarithmic compression at the pixel level [14]. This can achieve 5-6 orders of magnitude, but requires very precise circuitry [15]. More recently, Panasonic introduced a sensor with a claimed dynamic range of 140 dB, or 10,000,000:1 [16]. This is achieved by rapidly capturing three exposure-bracketed images which are combined on-chip using a feedback circuit attached to each pixel. A sample photo of an incandescent light bulb captured using exposure times of 1.5μS, 150μS, and 15mS, all captured within a single frame period, clearly show the bulb filament as well as the text on the bulb. The chip demonstrated has a low resolution of 177×144 and is aimed at automotive and security applications. A similar design has been released by Aptina Imaging [17]. For a more thorough review of HDR sensors, see [8, 15, 18, 19].

2.1.3 HDR Cameras

Grass Valley cameras created the Viper Filmstream camera, which converts 12-bit linear CCD outputs to a 10-bit per channel logarithmic format (part of the Filmstream interface specification) [20], producing about three orders of magnitude [30]. The camera has since been discontinued. Panavision offer a similar video camera which outputs 10-bit

log encoded data. Both of these cameras fall into medium dynamic range by offering a slight expansion of dynamic range while falling short of the ranges that can be captured with the techniques that will be discussed in Section 2.2.1.

SMaL Camera Technologies produced a camera called the Ultra-Pocket, based around their custom IM-001 CMOS sensor. The sensor uses SMaL's Autobrite technology to allow adjustment of individual pixel sensitivity, allowing the camera to capture about four orders of magnitude. At 640×480, the camera's resolution was a limiting factor for photographic applications [30]. SMaL has since been acquired by Sensata Technologies, who are using Autobrite in vehicle sensing systems such as rear-view cameras [21].

German company SpheronVR makes two HDR panoramic cameras, the Spherocam and the Spherocam HDR, with claimed dynamic range capture capabilities of 12 f-stops (three orders of magnitude) for the first and 26 f-stops (close to eight orders of magnitude) for the HDR model [22,36]. The image is captured by rotating a vertical line-scan camera 360° in the horizontal plane while capturing vertical lines (presumably with the camera set to auto expose), then assembling the vertical slices into a panorama. The recent models produce up to 50 megapixel images. While the capture process is one-shot in the context of the Spheron system, it is important to note that capture can take 15-30 minutes at full resolution [30].

Canadian company Point Grey Research produces the Ladybug line of spherical cameras, with current models Ladybug2 (used in our research) and Ladybug3, as well as recently announced model Ladybug5. These cameras incorporate six sensors into one housing, covering 75%-80% of a full sphere depending on model. The Ladybug2 offers two HDR capture options, which are discussed in detail in Section 3.1.

Recently, consumer grade cameras have begun to appear with HDR or pseudo-HDR functionality. In addition to the Fujifilm Super-CCD cameras, the Pentax K7, Casio Tryx, and Sony Alpha 500 and Alpha 550 cameras all advertise HDR functionality. In reality, they take 2-3 exposure-bracketed shots in quick succession, which are then combined and tonemapped in camera, so the final output is not HDR, though it can visually improve details in highlights and shadows. Apple's iOS operating system now allows a similar process on the popular iPhone.

While all of the HDR methods discussed here offer some improvements over traditional LDR capture, they all have their limitations. The specialized sensors discussed first have not been integrated into consumer cameras to date, aside from the Fuji Super-CCD which appears to only offer a slight improvement in dynamic range rather than true HDR. The specialized cameras discussed second typically have very particular applica-

tions; automotive imaging for the SMaL cameras, and high quality panoramas where capture time is not an issues for the Spheron Cameras. The consumer cameras discussed last do not even typically output HDR images, instead tonemapping them on-board. Unless otherwise noted, the rest of our discussion on the capture of HDR images will relate to the exposure-bracketing technique, which will be expanded upon in the coming sections.

2.2 HDR Fundamentals

The basic concept of HDR images is that they can contain a range of values much greater than traditional graphics formats. While traditional graphics formats (e.g. JPEG, Bitmap, PNG, etc.) store images as integers, typically 8-bit, HDR images are stored in floating point format allowing for a much greater range of values. There are two common ways to generate HDR images: artificially, using rendering approaches from computer graphics such as radiosity and ray tracing [30, Ch.4], or using techniques for combining exposure-bracketed images. Since this thesis is primarily concerned with the latter technique, we will use the term "high dynamic range imaging" and variants thereon to refer to the process of compositing HDR images from a series of low dynamic range (LDR) images. This section outlines the fundamental or naive HDR compositing process and the associated task of photometric calibration.

If we consider the images of Figure 2.1, no single image captures the full range of the scene. This is a scenario familiar to photographers, and typically requires some compromise, choosing the portion of the scene we most want to capture, and sacrificing the rest of the scene to over- or under-exposure. However, we notice that all parts of the scene are properly exposed in at least one image, which suggests that they could be combined to keep the best-exposed parts of each image. This could be done manually, and some early multi-exposure photography techniques did involve manually splicing negatives [1]. More recently, "exposure fusion" approaches have been proposed to automate the splicing of digital LDR images to directly composite another LDR image composed of the well-exposed portions of the input images [27, 37] and showing the details of all parts of the scene. However, another approach is to recognize that the knowledge of how the camera maps from real-world irradiance values to digital pixel values, combined with the exposure settings used, would allow us to combine the images into a single image in which pixel values are linearly related to irradiance, and the dynamic range is effectively only limited by the data type of the image storage format, e.g. a 32-bit float. This process is

outlined next.

2.2.1 HDR Recovery from LDR Images

To recover an HDR image from differently-exposed LDR images, the images need to be normalized to some common intensity domain, after which they can be averaged together. If the camera's overall transfer function, which converts irradiance at the sensor, E, to digital pixel values, Z, is perfectly linear, the normalization is as simple as dividing all images through by their exposure times (assuming all other exposure-affecting settings are held constant).

Usually though, a camera's mapping from radiance to digital pixel values consists of multiple steps, one or more of which may be non-linear as shown in Figure 2.2. Therefore, we must first recover the camera response function g, which maps Z back to exposure values, $X = E\Delta t$. Then, the exposures can be normalized by dividing by Δt before averaging together. The one remaining question is how to average them. Since both film and CCDs have an operating range outside of which they do not gather reliable information, we want to down-weight the contribution of values collected near and at the extremes of this range. This is done by introducing a weighting function so pixels in the unreliable portions of the camera's range contribute less to the final composite. Weighting functions are discussed in more detail in Section 2.2.2.

(a) (b) (c)

Figure 2.1: Images 2, 5 (auto-exposed), and 8 from a 9 image exposure-bracketed sequence.

In one of the earliest modern approaches to HDR recovery from LDR images, Madden proposes capturing exposure-bracketed images starting with a dark exposure in which no pixels are over-exposed and progressing geometrically through exposure times until no pixels are under-exposed [38]. A completely linear camera response is assumed, and pixel values are independently drawn from single input images. Two disadvantages of using a

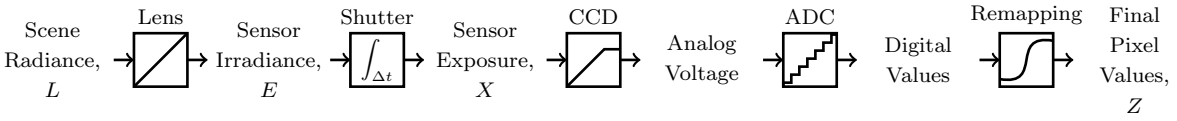


Figure 2.2: Camera transfer function block diagram. Adapted from [2]

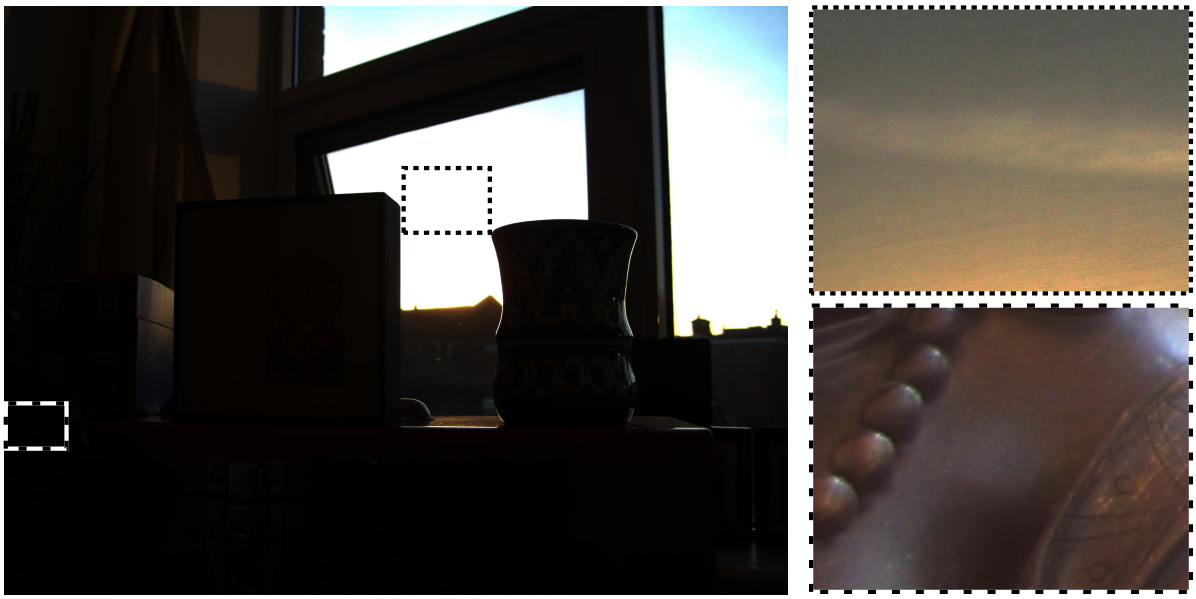
single image to obtain a pixel's HDR value are increased noise, detailed in Section 2.2.2, and discontinuities that can appear in the final image when transitioning from one input image to another.

Mann and Picard [26] outlined a method for recovering the response curve of an imaging system and combining exposure-bracketed images using a weighting function. Their arguments were based primarily on the properties of film, and used a restrictive gamma function, $Z = f(E) = \alpha + \beta E^{\gamma}$ as a model. To weight the images, they used the response function's derivative, arguing that portions of the response curve that were changing more rapidly indicated greater film sensitivity, which in turn indicated greater reliability in the measurements. While their work established the principles of HDR recovery, its basis in film means it does not generalize well to digital cameras.

More recently, Debevec and Malik [2], Mitsunaga and Nayar [6], and Robertson et al. [7] have proposed more generalized and robust techniques for recovering camera responses and combining LDR images into an HDR image. These techniques are reviewed in detail in Section 3.4.1. Finally, it should be noted that some graphics formats output by consumer-grade digital cameras, notably JPEG, include the tone curve in the meta data. The tone curve is the curve used by the camera to convert irradiance to digital values and for such cameras it is effectively the response curve. Current HDR software can use these curves to circumvent the calibration process [39]. However, with research-oriented cameras like the Ladybug2, this meta data is not typically available.

Discussion on HDR Recovery

Figure 2.3 shows the HDR image resulting from combining the images of Figure 2.1 using the method of Robertson et al. We notice immediately that very little detail is visible when the large dynamic range of the image is linearly scaled into the 8-bit range directly displayable on most monitors. However, if we just take a small window anywhere in the image and scale it into the displayable range, we can see that the full range of the scene has been captured. The process of compressing an HDR image into a displayable



(a) (b)

Figure 2.3: An HDR image created from the LDR images of Figure 2.1. a Entire image linearly compressed for display, showing loss of detail throughout. b Two independently compressed portions of the image, showing that well-exposed information is available throughout the HDR image.

range in such a way that the details are preserved is referred to as tonemapping and is discussed in Section 2.5. A false-colour visualization of the same result is shown in Figure 2.4, where we can clearly see the large range of intensities recovered.

While the discussion so far has focused on changing the exposure via the exposure time, it can also be varied using gain/ ISO settings. Regardless of the particular choice of response curve recovery method and weighting function, the final product of the process described in this section will be referred to as the naive HDR result in that it does not factor in motion of the camera, motion in the scene. In Section 3.5 we add a few details to the naive creation process

We also note that, while combining images of differing resolution (i.e. for a super-resolution approach) or images that are not precisely aligned would result in blurring of many or all features in the images, we only consider images of a fixed resolution, captured by a stationary camera.

Figure 2.4: False-colour visualization of the intensity of an HDR image with approximately 77,000:1 dynamic range. The scale shows intensities in orders of magnitude or $\log_{10}$ units relative to the darkest point in the scene.

2.2.2 Comparison of Weighting Functions

Various weighting functions have been proposed in the existing literature, and several are described in Section 3.4.1. Briefly, Debevec and Malik [2] propose a triangular hat function, Mitsunaga and Nayar propose the recovered response divided by its own derivative [6], and Robertson et al. propose a Gaussian-like function [7]. Figure 2.5 shows several of the weighting functions discussed. Additionally, Reinhard et al. advocate multiplying the Mitsunaga-Nayar weighting by a broad hat function so as to further down-weight the contribution of very over-exposed pixels [30].

Noise Removal Properties of HDR Imaging

In addition to the benefits of increased dynamic range, the multiple-image merging techniques discussed so far have the added benefit of reducing noise. Noise is particularly

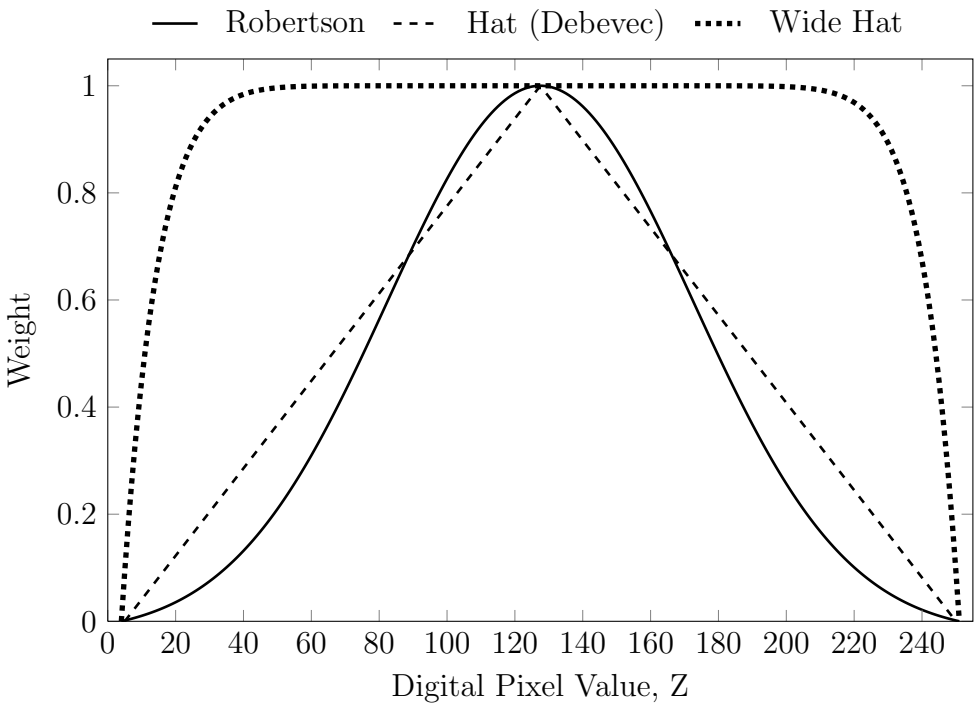


Figure 2.5: Some common HDR weighting functions

obvious in dark regions and/or at high ISOs, as shown in Figure 2.6a and 2.6b. However, since most noise sources are well approximated by symmetric statistical distributions about a mean, a simple averaging of multiple captures of the same image can greatly reduce the effects of noise, even for non-HDR imaging. This is shown in Figure 2.6c, where 9 shots of the same scene were averaged together. Since the naive HDR process uses a weighted average to combine the input images, the same effect will tend to reduce the noise in HDR images, and is a benefit of methods using a weighted average [2, 6, 7] over earlier methods taking each pixel's HDR value from a single input image [38].

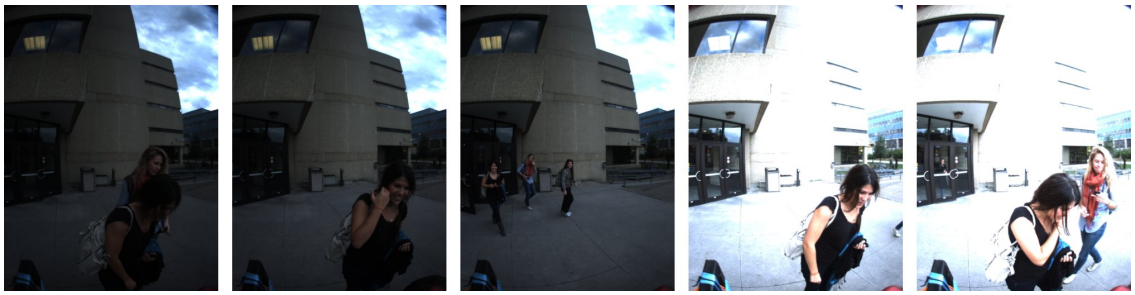
2.3 Ghost Removal

The discussion in Section 2.2.1 leads to an obvious question: given that the exposure-bracketed images are captured over some period of time, what happens if objects in the scene move? Since the naive HDR image is a weighted average of the exposure-normalized input images, if the objects were moving swiftly and continually throughout the capture period, the result is distinct faded copies, leading to the HDRI term "ghosting." Ghosting

Figure 2.6: Noise removal by averaging multiple noisy captures. a An image captured at 1600 ISO. b Zoom on a dark portion of the image. c Zoom on an image created by averaging 9 shots of the scene taken with the same settings.

can result from people, animals, and vehicles moving quickly through the scene. If the objects stayed in one place but moved or deformed during capture, this manifests as blurring. This type of ghosting commonly results from foliage blowing in the wind, waves on water, and cloud movement, and can also result from a person standing still while moving their limbs. An example of ghosting is shown in Figure 2.7.

There have been numerous approaches to the ghosting problem proposed in the literature, and these mostly fall into one of two categories. In the first category are optical flow-based approaches [27–29] in which optical flow algorithms are relied upon to "warp" the pixels of moving objects in each image so that they are aligned with respect to one



(a)

(b)

Figure 2.7: An example of ghosting. a Sample frames from a 9-image set with scene motion. b HDR result, tonemapped with a surreal tonemapper (Section 2.5) to better show the resultant ghosting in the near foreground, in front of the doors, and in the clouds.

reference image. Once this is complete, the naive HDR approach can be applied and the result should be a ghost-free scene in which high dynamic range is preserved in all portions.

In the second major category, which we will refer to as change detection, some variation on classical machine-vision style change detection, with adjustments to account for the changing exposure, is used to detect which portions of the scene have changed [30–32]. These portions of the naive HDR result are then replaced with information from one or more individual exposures in such a fashion as to reduce or eliminate ghosting.

A third less studied though important category approaches ghost removal as a background modeling challenge; one approach has been published which does this specifically for the purpose of ghost removal [40], and some authors of background modeling approaches for LDR images have demonstrated the application of their methods to HDR imaging [33, 34]. This section provides an overview of key papers within these classes of deghosting, and a summary and comparison of the methods is provided in Table 2.1.

2.3.1 Optical Flow-Based Approaches

One popular approach to ghost removal for HDR images, and the first to appear in the literature, is the optical flow approach. This typically involves choosing one reference image from the series of exposure-bracketed images (the "stack"). Then, for each of the other images, an optical flow field can be calculated to warp the pixels of that image so that they re-align with those of the reference image. Bogoni [27] uses a global affine transform followed by local dense optical flow to register images, but these are then combined through an exposure fusion process so we will not provide further details here. Kang et al. [28] use a combination of global homography and local dense optical flow to register and de-ghost images. Zimmer et al. [29] use a modern energy-based optical flow. These approaches are detailed below.

Combined Global Homography and Local Optical Flow

Kang et al. [28] proposed one of the first HDR deghosting methods. Their approach is specifically aimed at HDR video, and coincidentally used the first model of Ladybug camera. Since their goal is full frame-rate video, they only consider HDR bracketing with two brackets, and set their camera to alternate between these two settings frame by frame. Then, for a given frame, their goal is to warp the previous and next frames to align with it before applying the naive HDR approach.

Creating an HDR image in the place of a given long exposure, L, requires simulating short exposure information from the previous frame, S^-, and the next frame, S^+ so that for any pixel in the image we have both high and low exposure information. As a result, highly accurate pixel correspondences are needed. The authors introduce a scheme for warping S^- and S^+ to align with L, including provisions for when S does not provide good correspondences with L.

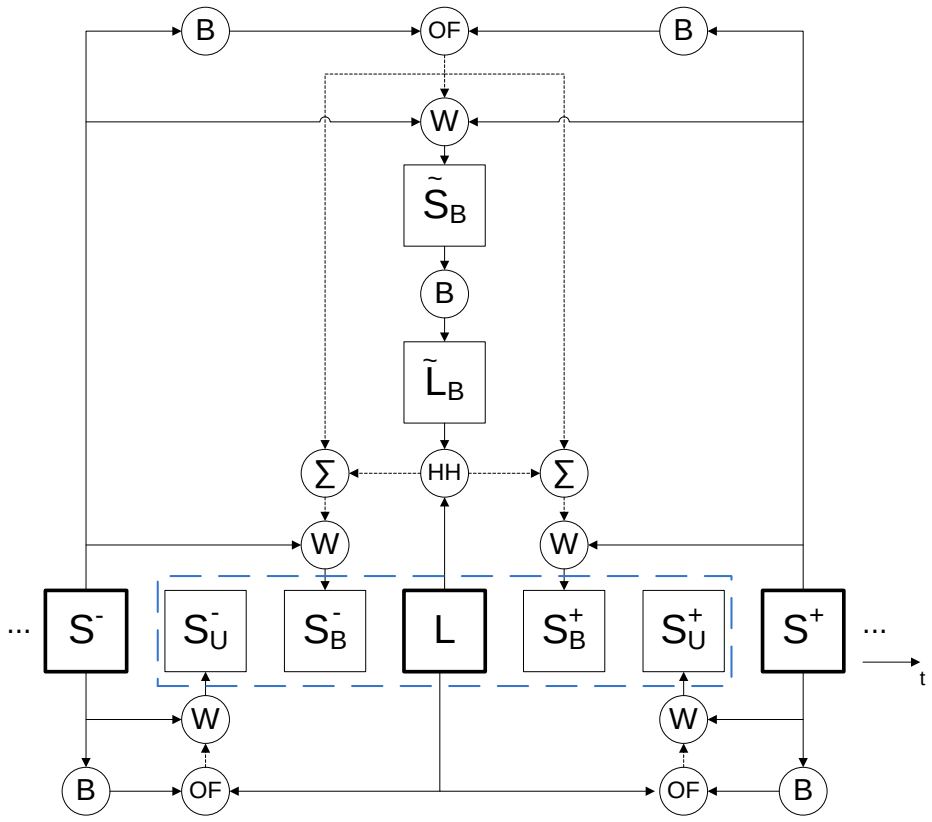


Figure 2.8: Ghost removal by Kang et al.'s optical flow method. S and L are short and long frames, bold boxes are the original frames, and the dashed box outlines the reference frame L and the synthesized versions of the S frames which are used to produce the HDR image. B, OF, HH, and W represent intensity Boosting, Optical Flow, Hierarchical Homography, and image Warping respectively. Dashed lines are motion vectors, and solid are image data. Adapted from [28]

In the case that S provides good correspondences with L and therefore a good optical flow field, unidirectional simulated frames S_U^- and S_U^+ are created by first "boosting"

the S frames' intensities then calculating their optical flow with L. The boosting is accomplished by first converting them into radiance images using the camera response function, then using the inverse response function to create "virtual" exposures of them at the same exposure as L. This is similar to the "virtual photography" technique described by Debevec and Malik [2]. Once the boosted versions of S are available, S_U^- and S_U^+ are calculated using a custom motion estimation algorithm, described below. In the case that S^+ and S^- do not share enough well-exposed pixels with L to provide a good flow field, the authors create bidirectional images S_B^+ and S_B^-. This is done by first calculating the bidirectional flow field between S^- and S^+ directly, ignoring L. Then, S^- and S^+ are warped to the mid-point of this field to create $\tilde{S}_B$ which is theoretically now registered to L. However, due to camera movement and the fact that L was not actually considered in the creation of $\tilde{S}_B$, the registration may be imperfect. So, a homography is calculated between $\tilde{S}_B$ and L using a hierarchical homography scheme. This homography is then combined with the flow fields just calculated to create the two bidirectionally simulated images, S_B^+ and S_B^-. This process is shown in Figure 2.8.

The motion estimation between L and S involves first calculating an affine transform to "globally" register S to L, then calculating a dense flow field to "correct" the affine transformation. The authors use a modified Laplacian pyramid version of Lucas-Kanade optical flow for the dense field, and add enhancements to account for degenerate cases. For bi-directional frames S_B, they calculate a bi-directional flow so that each pixel in the simulated frame has two vectors, each pointing into either S^+ or S^-. For unidirectional frames S_U, the motion estimation is used to produce unidirectional vectors relating S to L so that the simulated frame S_U is generated directly from S.

Regardless of the choice of bidirectional or unidirectional warping schemes, the end result of the warping process is three differently exposed images, L and the warped versions of S^+ and S^-. The authors combine these images using Mitsunaga and Nayar's method. The entire process can be repeated with an S frame as the reference by swapping S and L in the above discussion and rearranging boosting stages as necessary. While the two S frames may seem redundant from an HDR point of view, multiple images at the same exposure can in fact reduce noise, and in this case, they likely make the registration more robust. The authors also outline how to apply this process to more than three exposure-bracketed images by applying the registration/ warping process to adjacent pairs of images and then combining the warping of subsequent pairs of images to bring the furthest images into alignment with the reference.

Super-Resolution HDR Using Energy-Based Optical Flow

In a more recent work, Zimmer et al. [29] re-visit optical flow as a simultaneous HDR registration/ deghosting method with the addition of super-resolution. They adopt a modern energy-based optical flow with modifications to allow it to operate on differently-exposed images. Unlike [28], their method does not need the camera response function in order to perform the alignment. They additionally leverage the sub-pixel accuracy of their optical flow algorithm to perform simultaneous super-resolution (SR) and HDR compositing, calling the approach SR-HDR. While work previous to theirs proposed SR-HDR solutions using energy minimization [41, 42] and showed that combined SR-HDR approaches worked better than treating the SR and HDR problems separately, it did so with poor priors that weaken the overall results.

The authors claim two novel contributions. First, they perform HDR alignment using modern energy-based optical flow, giving dense displacement fields with sub-pixel accuracy and handling the necessary exposure changes using a gradient-constancy assumption that is independent of global illumination changes. Second, they use the sub-pixel displacements from their optical flow as inputs to a simultaneous super-resolution process using a robust data term and a novel anisotropic smoothness term that smooths the final image while preserving sharp edges.

The authors provide compelling results for scenes with camera shake and scene motion. However, their example of scene motion only involves cloud motion, and they do not show in their paper more traditional ghosting scenarios, i.e. discrete moving objects in the scene such as people and cars. In their supplementary material [43], they do offer one example of the latter type of ghosting, but this is also their example of their algorithm failing, so it is unclear how well it generalizes to this type of ghosting. This highlights a common pitfall of optical flow, in that it struggles with large motions of discrete objects. In Section 6.1 we discuss this method further.

2.3.2 Change Detection Approaches

Another class of ghost removal methods is what we will refer to loosely as change-detection. In these approaches, some form of change detection is applied to the exposure-bracketed image stack to detect which portions of the images have been corrupted by motion. Once these portions have been detected, they can be replaced with information from some subset of the input images so that the final HDR result does not display ghosting.

In basic change detection, which is often used in machine vision applications such as product inspection and motion detection for surveillance applications, movement is detected by taking the absolute difference of two images of the same scene taken at different times. The absolute difference image is then thresholded, typically either using a fixed percentage threshold so that, for example, a 20% change in a pixel's value indicates it has seen a moving object, or adaptively using a method such as Otsu's method [44]. The result is a binary image with a value of 1 for pixels that have seen motion and 0 for those that have not. Since various noise factors can result in broken up regions, the result is often processed with morphological operators to clean it up, bringing together proximal disconnected regions and rejecting regions with areas below some threshold, e.g., rejecting regions that are less than 0.1% of the overall image area. A more in-depth look at change detection and the related challenge of background modeling is discussed in Section 2.3.3.

However, when applied to exposure-bracketed images, straightforward change detection does not work since the changing exposure creates large global illumination changes that result in many static parts of the scene being detected as moving. So, variations on change detection have been developed to handle the large and continual illumination changes resulting from exposure-bracketing.

Variance-based Ghost Removal

One of the earliest change detection style deghosting methods is from Reinhard et al. [30, Ch.4] who propose using a weighted variance over the image stack as an indicator of motion. A block diagram is provided in Figure 2.9 where, for this method, the motion-corrupted regions are detected using a weighted variance.

As the weighted average across the stack is accumulated to create the naive HDR result, a weighted variance image can be accumulated as well. The weighted variance is defined for a given pixel as the weighted sum of squares across the stack divided by the square of the weighted average, all minus one. That is, the variance image V is defined as [45, Ch.10]:

$$V(i,j) = \left[\frac{\frac{\sum_{n=0}^{N} W_n(i,j) E_n(i,j)^2}{\sum_{n=0}^{N} W_n(i,j)}}{\left(\frac{\sum_{n=0}^{N} W_n(i,j) E_n(i,j)}{\sum_{n=0}^{N} W_n(i,j)} \right)^2} \right] - 1. \tag{2.1}$$

In addition to the weighting, this differs from the standard equation for variance in that both sides have been divided by the square of the weighted mean. The authors

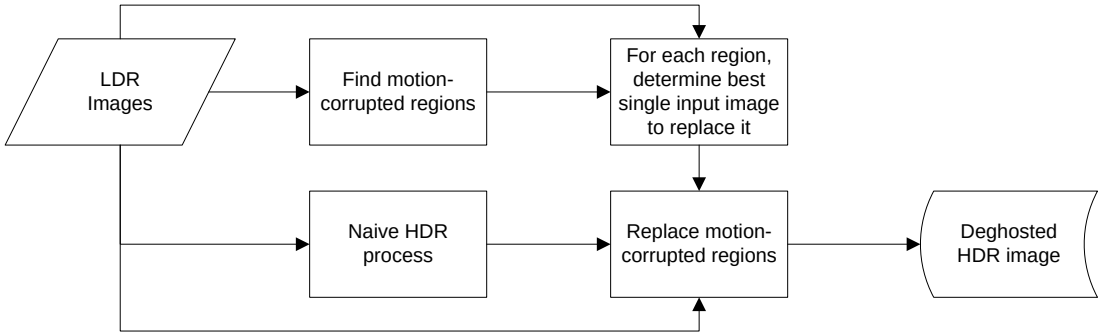


Figure 2.9: Block diagram for the variance-based and entropy-based deghosting methods. The input images are used to calculate a mask outlining regions that were corrupted by motion. Each region is then replaced in the naive HDR result with information from a single input image.

compute separate variance images for R,G, and B, then take the maximum across the three channels. Rather than threshold the full-resolution variance image at this point, they first reduce its resolution by a factor of 10 in each dimension to reduce computation (this is likely not necessary on current computers) before thresholding it at a variance of 0.18 so that any pixel with a greater variance is considered to have been corrupted by scene motion. They clean up the result morphologically, dilating the result then rejecting any ghosts smaller than 0.1% of the image.

Once motion-corrupted regions have been identified, the authors choose, independently for each region, the best exposure to replace that region, i.e. the replacement for that region is taken from a single input image. The best exposure for a given ghosted region is found by creating a histogram of the values in that region, rejecting the top 2% of values, then looking for the longest exposure that contains this 98% maximum within its valid range (presumably as dictated by the weighting function). Finally, rather than directly substituting values from this exposure, the values are mixed with those of the naive result using the variance as the mixing coefficient so that very low variance pixels are not replaced; this last step was found to reduce artifacts.

The authors show a sample result which is encouraging considering the simplicity of the approach. However, it does show breakup of some of the moving objects. Furthermore, the choice of best exposure for each ghost region leaves open the possibility of duplication of moving objects if a single moving object were responsible for multiple ghosting regions and also happened to represent the best exposure for multiple regions.

Entropy-based Ghost Removal

Jacobs et al. [31] present a method to automatically align and de-ghost HDR images. For alignment, they propose an improved version of the Median Threshold Bitmap (MTB) method introduced in [46]. For their change detection, they propose an exposure-independent measure based on entropy which does not require the camera curve. A block diagram is provided in Figure 2.9 where, for this method, the motion-corrupted regions are detected using the entropy measurement.

In the original MTB method, it is noted that thresholding a grayscale image at the median pixel value such that pixels above that value become 1 and below become 0 results in a binary image that is changed very little by differences in exposure. As a result, the MTB of two differently exposed but aligned images should look nearly identical. Images that are not aligned will have slightly different MTBs. Taking the exclusive OR (XOR) of the two MTBs for two images gives an indication of how misaligned they are. Adjustments to the x, y alignment of the images will affect the XOR of the MTBs; an adjustment that brings them into closer alignment will reduce the total difference (indicated by the sum of the XOR), barring local minima problems, which are largely precluded by the multi-resolution approach outlined below.

The problem then consists of finding the x, y offsets that will bring the second image into alignment with the first. The offsets are found using a multi-resolution (pyramid) minimization process in which the lowest resolution level is aligned first, and the resulting alignment is used as a starting point for aligning the next level. When the top level alignment is complete the result is a total integer x, y offset that aligns the images to within a single-pixel precision (as opposed to sub-pixel).

The grayscale images required as initial inputs to the algorithm can be approximated from the green channel, or from a proper conversion to grayscale. Provisions to handle noise issues when many pixels are near the median are also proposed. In [31], Jacobs et al. modify the MTB algorithm by adding rotation as a third search dimension.

For their ghost removal approach, Jacobs et al. note that the entropy of an image is independent of global illumination changes, and in fact independent of any injective function. They introduce a definition of the entropy of pixel X as

$$H(X) = -\sum_{x} P\left(X = x\right) log\left(P\left(\left(X = x\right)\right)\right), \tag{2.2}$$

where $p(x) = P\left(X = x\right)$ is the probability that a given pixel has the value x, and is given by the M-bin histogram of the image normalized so that $\sum_x p(x) = 1$. The authors note

that 1) while a higher entropy indicates there are more different values in an image, the actual intensity of the values does not matter, 2) the order of pixel values does not affect the result, and 3) scaling the values in the image does not affect the entropy measure.

For each input image, a corresponding entropy image, H_n, is calculated. Within this image, the value for a given pixel (i, j) is calculated by evaluating expression 2.2 over a small window around the pixel so that the pixel value in H_n becomes a measure of the local entropy around that pixel in that image. The result is a series of images, $H_n, n \in [0, N-1]$, one for each of the N LDR input images. These are then combined as a weighted entropy difference to create an "uncertainty image" UI as:

$$UI(i,j) = \sum_{m=0}^{N-1} \sum_{n=0}^{n<m} \frac{v_{mn}}{\sum_{m=0}^{N-1} \sum_{n=0}^{n<m} v_{mn}} h_{mn}(i,j), \tag{2.3}$$

$$h_{mn}(i,j) = \left| H_m(i,j) - H_n(i,j) \right|, \tag{2.4}$$

$$v_{mn} = min\left(W_m(i,j), W_n(i,j) \right), \tag{2.5}$$

where W is a weighting function (see Section 2.2.1) used to discard over-/under-saturated regions which do not tell us anything about entropy. The authors recommend a narrowed hat function with lower and upper limits of 0.05 and 0.95 for pixel values normalized to $[0, 1]$.

The final interpretation of UI is similar to that of VI from Section 2.3.2: high values indicate areas which have seen movement. The authors use a 200-bin histogram and a 5x5 window in the entropy calculation, and threshold UI at 0.7 to create a binary image UI_T indicating areas corrupted by motion. This is cleaned up with standard morphological operations. The final HDR image is generated by replacing each ghosted region with values from the input image having the least saturation for that region.

Extending Dynamic Range of a Reference Image

Gallo et al. [32] first choose a reference image from the stack, then extend the dynamic range of this image on a patch-by-patch basis by including information from the other images in the stack if they are consistent with the reference image over that particular patch. This approach has two main advantages. First, any one input image is self-consistent, meaning it captures a single instant in time and therefore does not contain duplicates of any moving objects. Therefore, if the dynamic range of one image can be

extended with only consistent parts of the other images, no duplication will result, a guarantee not offered by the two previous techniques. Second, the previous methods find large ghosted regions and replace the entirety of the region with a single exposure, which can result in a reduction in dynamic range in those regions if they are sizable. Extending the dynamic range within a ghosted region with consistent data from other shots can still achieve high dynamic range in ghosted regions. A block diagram for this method is provided in Figure 2.10.

The authors assume that the input images are aligned and the camera response function g is known. The reference image can be chosen manually by the user, or using an automated procedure as follows. First, saturated pixels in all the input images are found. Morphological operators are used to discard very small saturated regions regions since they are assumed to still have enough texture in their local neighbourhood that they can be compared to other images. The remaining regions are summed for each image and the image with the fewest saturated pixels is taken as the reference.

The next task is to extend the dynamic range of the reference image patch-by-patch. The image is broken down into a 40×40 grid of patches; patches can be of arbitrary shape, i.e. the output of a segmentation routine, but the authors claim good results with square patches. Now, for a given patch in the reference image r, being compared to another image n in the stack, the pixel values in n are linearized through $g(\cdot)$ and plotted against their corresponding linearized values from r. Since the exposure, $x_{i,n} = E_i \Delta t_n$, of pixel i is linearly related to E_i by the exposure time, the pixel exposures in n should be related to those in r by

$$g\left(\mathbf{z}_n\right) = g\left(\mathbf{z}_r\right) \cdot ev_{rn}, \tag{2.6}$$

where ev_{rn} is the relative exposure of image n with respect to r. As a result, the pixel values in the plot will cluster around a straight line (barring all noise and saturation problems they would form a perfectly straight line).

However, if n has seen something different than r in this patch, then a sizable portion of the pixels will lie away from this straight line. As such, parallel lines can be introduced on either side of the expected straight line, at some threshold distance. Pixels lying outside these thresholding lines are considered outliers indicating movement. Then, a *ghosting value* can be calculated as the ratio of the the number of outliers to the total number of pixels in the patch.

Two thresholds, one for the distance of the thresholding lines from the expected line, and the other for the ghosting value cutoff, can be tuned by the user as needed. Lower

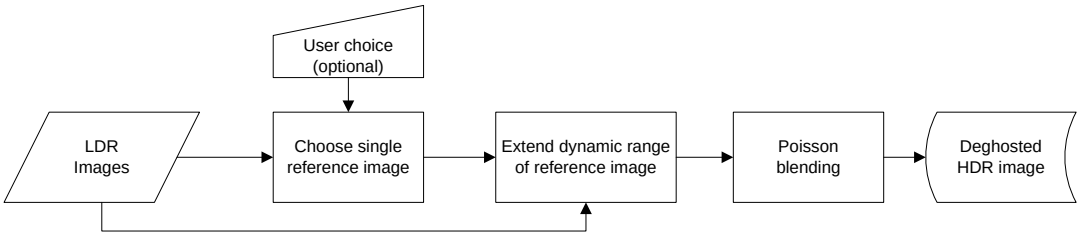


Figure 2.10: Extending the dynamic range of a reference image. A single, inherently ghost-free input image is chosen automatically or manually, then divided into a 40x40 grid of square blocks. On a per-block basis, the dynamic range is extended with information from other input images if they are consistent with the reference image. Poisson blending is used to reduce artifacts at edges of blocks.

thresholds will remove more subtle ghosting effects but will also reduce dynamic range more in ghosted areas. Finally, since the image is inspected on a patch basis, artifacts can appear in the form of sharp transitions between adjacent patches when they have taken their dynamic range extensions from different subsets of the input image stack. To reduce this effect, the authors perform Poisson Blending, detailed in their paper, but even so, discontinuities between patches can still occur in some regions.

2.3.3 Ghost Removal by Background Modeling

The challenge of modeling the background of a scene from a series of images, which is commonly desired in both machine vision (surveillance) and computer graphics (matting) applications, is closely related to the HDR deghosting problem. The problem can be thought of as finding the set of pixel values for a final composite image whose distance from the camera is maximum, i.e. for a given pixel, its value should be taken from the input image in which it saw the most distant object. Since most applications lack depth information, this is approached by removing all transient objects in the final scene.

While background modeling techniques are usually aimed at LDR applications, Khan et al. [40] use Kernel Density Estimation specifically for HDRI to perform probabilistic deghosting that, under ideal conditions, acts as a background estimator. Granados et al. [33] and Chen et al. [34] model the background using graph cuts to minimize an energy function, and show extensions of their estimation algorithms from LDR to HDR. We detail the three approaches below after providing a brief overview of traditional background modeling techniques.

Background Modeling and Foreground Segmentation

Background modeling is an important first step to segmenting foreground objects in many computer vision applications, notably surveillance applications in which we want to detect people, cars, and other transient objects in a scene, and machine vision applications like automated inspection in which we want to segment a product on an assembly line from the assembly line itself. Once a background model has been created, it is compared to subsequent frames to reveal what portions of the image have changed and should therefore be considered part of the foreground. Typically, some training stage is involved during which the background model is constructed, followed by an on-line stage where segmentation is performed on a frame-by-frame basis. In some applications, the background is also updated during run-time to account for slow changes over time. The following provides a brief overview of existing background modeling approaches and discusses challenges in applying them to HDR imaging. See [47, 48] for more detailed reviews.

The simplest background modeling approach is to assume that the previous frame represents the background. We can then take the absolute difference of the current and previous frames, and choose a threshold for the result, above which pixels are considered foreground. This can be improved by taking the average of previous frames as the background to iron out mild lighting variations. Both of these methods tend to be too simplistic to single-handedly perform background modeling in complex situations or when some components of the background undergo periodic changes such as a tree leaf blowing in the wind. The following is a selection of approaches proposed to provide more robust background modeling and segmentation.

Wren et al. [49] model the background on a per-pixel level by fitting a Gaussian probability density function (PDF) to the previous n samples so that only recent information is considered in modeling the background rather than the whole history of the scene. Since a Gaussian PDF is described parametrically by its average, μ, and standard deviation, σ, computation and memory can be saved by updating these parameters on-line, removing the need for a buffer of previous values. In the next frame, a pixel with value I_t is classified as foreground if $|I_t - \mu| > k\sigma$, where k is a threshold. This method can be improved by selectively updating the background only with pixels that were classified as background in the current frame. The method of Wren et al. would typically train over 30-1,000 frames to build a robust background model [50, Ch.9].

Stauffer and Grimson [51] model the background for a pixel using a *mixture* of Gaus-

sians, where each of the K Gaussians models a different previously observed state of the background. This allows the algorithm to model behavior such as a leaf moving periodically in the wind, which should be considered part of the background. Power and Schoonees [52] provide a tutorial on this approach, including details on initializing the Gaussian parameters. For each mode, μ and σ are maintained using on-line updates. To classify the pixel in the current frame, it is compared to the known distributions and is considered a "match" for a distribution if it is within $\pm 2.5\sigma$ of that distribution's peak. When more than one match is found, the distribution with the highest peak is chosen. If no match is found, the pixel is classified as foreground in the current frame, and the distribution with the lowest peak in the model is replaced with a new wide Gaussian centered on the new observation. This last step allows newly stationary objects to become part of the background model over time.

Toyoma et al. [48] use three levels of change detection in their algorithm: pixel-level, region-level, and frame-level. At the pixel level, they use a Wiener prediction filter to predict an expected value for the pixel in the next image based on previously observed values (the authors use 50 previous observations). Any pixel more than $4\sqrt{E[e_t^2]}$ different from its expected value is classified as foreground, where $E[e_t^2]$ is the expected squared prediction error. The authors claim that the predictor works well for periodically changing pixels. For region-level segmentation, the authors observe that large homogeneously coloured moving objects will often only produce foreground pixels at the edges. They introduce a custom method for propagating foreground classifications inward, though its description is unclear and it appears to be based only on 4-neighbour connectivity and one iteration of propagation, so it would likely not work to fill in large objects. For frame-level detection, the authors note that a sudden global illumination change due to an event like a light switch being turned on looks, at the pixel-level, like the appearance of a moving object, and can only be detected on a whole-frame level. They therefore maintain multiple sets of background models for various lighting situations, and when a global illumination change is detected, they switch models, choosing the model which best describes the new illumination.

Elgammal et al. [53] use Kernel Density Estimation (KDE) to model the background for a given pixel. Kernel density estimation is described in more detail in the following section, but in the current context, it can be understood as a smoothed and continuous histogram. This is advantageous in a scenario where we have a limited number of samples of the background; the histogram, being discrete, will often poorly model the PDF and lose the tails of the distributions [47]. The KDE is calculated from the previous n

observations, kept in a FIFO buffer; this requires more memory than parametric models such as [51], but updating the model simply requires adding the latest observation to the buffer. The current pixel value is classified as foreground if $P_r(x_t) < T_h$, where $P_r(x_t)$ is the KDE for the current value based on the previously observed values and will be lower if pixel is more different from the observed values. See [53] for details.

Lo and Velastin [54], as part of a larger system to detect pedestrian congestion on subway platforms, use the median value of the previous n frames as the background model. This has the advantage of being very simple and quick to calculate, though it does require a memory buffer.

Kim et al. [55] model the background by storing a pixel's history in codebooks, where a codebook is a series of volumes in a 3D colour space in which the pixel has been observed previously [50, Ch.9]. During training, if the current value lies in one of the existing volumes, it is classified as background. If it lies very near one, the volume is expanded to encompass the new value. If it lies far from any existing volumes, a new one is started at that value. In the runtime stage, a pixel is classified as foreground if its value lies within a threshold distance of one or more volumes in its codebook. The authors provide a metric to compare colour and intensity separately, but these can still be thought of as forming one volume whose particular shape depends on the colour space used; [50, Ch.9] describes its use in RGB, where they form boxes.

From an HDR deghosting perspective, there are a few important things to note about these background modeling techniques:

- **Periodic Background Behaviour** In most of these algorithms, it is desired to have the background model encompass any periodic motion of the background such as a leaf blowing in the wind which, for a single pixel, might mean alternating between blue (sky) and green (leaf), where both should be considered background. In a deghosting context, we do not want to consider both of these part of the background.

- **Learning Time** Most of these algorithms assume we have at least several dozen frames in which to learn the background model, and some papers cite up to 1,000 frames. In HDR imaging, we will rarely be working with more than 25 images, and often may be using as few 10[1].

[1]Even 10 frames is a lot by HDR standards. Many photographers work with sets of 3-7 images spaced at 1EV. However, due to the low RAW bit-depth of the Ladybug2 camera, we typically need at least 10 frames at 1EV to thoroughly cover the range of real-world scenes.

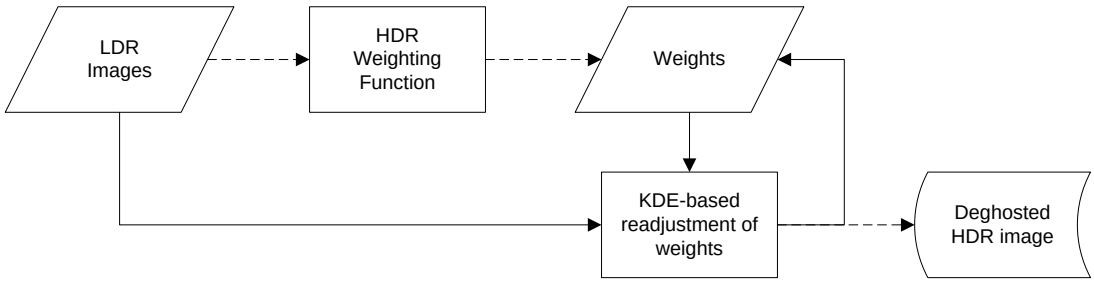


Figure 2.11: Iterative re-weighting by Kernel Density Estimation (KDE). Solid lines indicate interated steps, dashed lines indicate single-execution steps. The HDR weighting function initializes the weights. Then, KDE estimates the probability that a pixel saw the background in a particular input image; this is used to adjust the weight for that pixel-input image combination.

- **Lighting Variations** Most of the algorithms consider lighting variations in some form, but rarely on the scale and frequency that occurs due to exposure-bracketing.

Iterative Downweighting by Kernel Density Estimation

Khan et al. [40] propose a method of iterative re-computing of the HDR weights based on their likelihood of belonging to the background. Their likelihood of belonging to the background is in turn calculated using a simple non-parametric statistical process. For a given pixel, the naive HDR process could be seen as weighting the contribution of each exposure-bracketed image to that pixel's final HDR value based on its probability of being correctly exposed. This process can be improved upon by additionally weighting that image's contribution based on the probability that it has seen the background and not a moving object. One assumption made in this process is that, for a given pixel, the neighbourhood around it, defined by a small neighbourhood in pixel coordinates extended across the image stack, sees mostly background. A block diagram for this method is provided in Figure 2.11.

Since the weighting function is already known, the challenge is to find the probability of membership in the background class F. This is done using a kernel density estimator,

$$P(x|F) = \frac{1}{M}\sum_{m=1}^{M} K_{\mathbf{H}}(\vec{x} - \vec{y}_m), \tag{2.7}$$

where the authors choose a Gaussian density

$$K_{\mathbf{H}}(\vec{x}) = |\mathbf{H}|^{-\frac{1}{2}} (2\pi)^{-\frac{d}{2}} exp\left(-\frac{1}{2}\vec{x}^T\mathbf{H}^{-1}\vec{x}\right). \tag{2.8}$$

Vectors $\vec{x}$ and $\vec{y}$, $\vec{y} \in F$, are vector representations of the pixel values in a five-dimensional feature space, with three dimensions for colour and two for location in the image. The authors use the decorrelated $L\alpha\beta$ colour space[2], citing its perceptual uniformity [57]. $\mathbf{H}$ is a symmetric, positive-definite, $d \times d$ matrix; the authors choose the identity matrix, but another option is given in [45, Ch.10] as:

$$\mathbf{H} = \begin{pmatrix} \tilde{\sigma}_L(N_{i,j}) & 0 & 0 \\ 0 & \tilde{\sigma}_\alpha(N_{i,j}) & 0 \\ 0 & 0 & \tilde{\sigma}_\beta(N_{i,j}) \end{pmatrix}, \tag{2.9}$$

where $N_{i,j}$ is the neighbourhood about pixel (i, j) and $\tilde{\sigma}_L(N_{i,j})$ is the local weighted standard deviation of values along dimension L (likewise for α and β). The discussion above assume all pixels are initially considered equally part of the background, but does not factor in the effects of under-/over-exposure in the background membership calculation. This can be done by factoring the naive HDR weights of Section 2.2.2 into the probability calculation so that the probability that vector $\vec{x}_{ijn}$, for the pixel at (i, j) in image n, belongs to the background becomes

$$P(x_{i,j,n}|F) = \frac{\sum_{p,q,s} w_{p,q,s} K_{\mathbf{H}}\left(\vec{x}_{i,j,n} - \vec{y}_{p,q,s}\right)}{\sum_{p,q,s} w_{p,q,s}}, \tag{2.10}$$

$$(p, q, s) \in N\left(\vec{x}_{i,j,n}\right), (p, q) \neq (i, j).$$

If the assumption that the neighbourhood N sees mostly background is true, then a pixel seeing a moving object will get a lower probability than one seeing the background, even if their exposure is the same. The probabilities from expression 2.10 can therefore be used as weights in the naive HDR process. Finally, noting that the weight function shown in expression 2.10 is the naive HDR weight function, and that there is now a set of improved weights, the new weights can become the initial weights in a second iteration of this process:

[2]The $L\alpha\beta$ colour space should not be confused with the similarly-named $L^*a^*b^*$ (aka CIE 1976, CIELAB) space or the less common *Lab* (aka Hunter 1948) space. See [56, Ch.8] for a review of colour spaces.

$$w_{p,q,s,t+1} = w\left(Z_s(p,q)\right) P(\vec{x}_{p,q,s}|F), \tag{2.11}$$

where $w\left(Z_s(p,q)\right)$ is the initial weight at pixel (p,q,s). Iterating expressions 2.10 and 2.11 will cause ghosts to fade away; the process can continue until weights converge to within some threshold or a maximum number of iterations is reached.

In reality, the assumption that a neighbourhood sees mostly background is often violated, in which case this algorithm will converge toward the moving object's pixel values rather than the background, or weakly satisfied (e.g. a pixel only sees background in 6 out of 10 shots), in which case it will converge very slowly.

Graph Cuts with Entropy-Tuned Energy Function from Likelihood and Stationariness

Graph cuts can be used to minimize an energy function over a large graph such as an image. Given a series of input images, we want to find a set of labels equal in size to the input images, where the labels serve as an index into the input images based on model assumptions about the background. Then, an output image can be created directly using the labels in the graph cut result as indexes into to the input images.

Typically, an energy function, calculated over the input images, includes both a data term and a smoothness term. The data term encodes what we know about a specific pixel and while it may be calculated over a neighbourhood around that pixel, it only tells us the cost of assigning a particular label to that pixel. The smoothness term encodes what we know about the spatial relationship between pixels and is evaluated for pair-wise pixel labelings, i.e. if two pixels i and j are given the labels p and q, the combined cost of this labeling is given by the smoothness term.

Granados et al. [33] build on earlier work by Cohen [58]. Cohen proposed a method for background estimation using graph cuts and an energy function based on the assumptions that background objects are stationary and are more likely to appear than moving objects. For a two-part data term, Cohen penalizes (gives higher cost to) 1) low colour stationariness, which manifests as a large temporal variance in pixel value in a short time period, and 2) motion boundary inconsistencies which manifest as large differences in spatial gradient between two images. Granados et al. present an overall energy function as follows:

$$E(f_p) = \underbrace{\sum_{p \in \mathcal{P}} D_p(f_p)}_{\text{data term}} + \underbrace{\sum_{(p,q) \in \mathcal{N}} V_{p,q}(f_p, f_q)}_{\text{smoothness term}} + \underbrace{\sum_{(p,q) \in \mathcal{N}} H_{p,q}(f_p, f_q)}_{\text{hard constraint}} \tag{2.12}$$

where f_p denotes a labeling for pixel p, $\mathcal{P}$ is the set of pixel positions in the image, and $\mathcal{N}$ is the set of pairs of 4-connected neighbours (p, q) in $\mathcal{P}$. While Cohen's work is aimed at video, Granados et al. address non-time sequence images, in which there is no constraint on the time in which they are captured, though they must be captured in the same illumination conditions. Therefore, they propose a modified two-part data term $D_p(f_p)$ composed of a likelihood term, $D_p^L(f_p)$, and a stationariness term, $D_p^S(f_p)$:

$$D_p(f_p) = (1 - \beta(p))\, D_p^L(f_p) + \beta(p) D_p^S(f_p) \tag{2.13}$$

where β is a tuning parameter. The likelihood term is adapted from earlier work by Agarwala et al. [59] and penalizes values with a low probability across the input image set. The probability is approximated independently for each colour channel with a Gaussian kernel density estimator (see Section 2.3.3); since this requires independent probabilities in the channels, the decorrelated $L\alpha\beta$ colour space is used as in [40].

The stationariness term uses a linearized approximation to Cohen's Motion Boundary Consistency term to penalize motion boundaries that do not coincide with an intensity edge in the background image. Motion boundaries for a given image are approximated from the gradient of the difference of that image and the background; the difference of these motion boundaries and the gradient of the background in turn gives the cost:

$$D_p^S(f_p) = \begin{cases} \left\|\nabla M_{f_p}\right\|_2 - \left\|\nabla \bar{I}(p)\right\|_2 & \text{if} \quad \left\|\nabla M_{f_p}\right\|_2 > \left\|\nabla \bar{I}(p)\right\|_2 \\ 0 & \text{else,} \end{cases} \tag{2.14}$$

where $M_{f_p} = I_{f_p} - \bar{I}$ is the difference of the current image and the background image, $\bar{I}$, which is initialized as the average of the input images. Note that the likelihood and stationary terms are both in the range $[0, 1]$, unlike Cohen's equivalent terms, making them easier to tune relative to one another.

Granados et al. introduce an automated method for tuning β based on the entropy of a pixel's observed values. The observed intensities are assigned probabilities $\mathbf{P}_p(l)$ based on those already calculated in the likelihood term. Then, $\beta \in [0, 1]$ is calculated as the normalized entropy:

$$\beta(p) = -\frac{1}{\ln(N)} \sum_{l \in \mathcal{L}} \mathbf{P}_p(l) \ln\left(\mathbf{P}_p(l)\right), \tag{2.15}$$

$$\mathbf{P}_p(l) = 1 - \frac{D_p^L(l)}{\sum_{k \in \mathcal{L}} D_p^L(k)}. \tag{2.16}$$

where $\mathcal{L} = \{1, 2, \ldots, N\}$ is the set of labels of the input images. Entropy gives a measurement of the uncertainty remaining given the measurements available. When the entropy is high, this indicates there is very little information on a pixel's likelihood and therefore the likelihood *penalty* should be reduced regardless of its value. High entropy occurs when either the background is severely occluded (all images have uniform and *low* probability) *or* it is never occluded (all images have uniform and *high* probability). In the former case, we want to reduce the likelihood penalty since the statistics of the images tell us nothing about what the value should be. In such regions, the stationariness term should dominate. In the latter case, the likelihood penalty will be low anyway so β has little effect. When neither likelihood nor stationariness terms provide any information, the smoothness term, discussed below, is relied upon to propagate information from other regions.

Since the final result is a composite of the input images, transitions in labeling must occur at various points throughout the result. These labeling transitions should occur in regions where the images pertaining to each label are very similar. This constraint is captured by the *smoothness* term, calculated on pairs of neighbouring pixels, that "penalizes intensity differences along labeling discontinuities." The smoothness term is given as:

$$V_{p,q}(f_p, f_q) = \frac{\gamma}{2} \left(\left\| I_{f_p}(p) - I_{f_q}(p) \right\|_2 + \left\| I_{f_p}(q) - I_{f_q}(q) \right\|_2 \right). \tag{2.17}$$

This ensures that for neighbouring pixels p and q with candidate labels f_p and f_q, $f_p \neq f_q$, each pixel has a similar value in both images. The smoothness term is weighted by γ; increasing it decreases the number of labeling regions.

Since the smoothness term can be overshadowed by the data term, the authors introduce a novel hard constraint to enforce local consistency:

$$H_{p,q}(f_p, f_q) = \begin{cases} 0 & \text{if} \quad \min\limits_{l \in \mathcal{L}} \left(\left\| I_{f_p}(p) - I_l(p) \right\| + \left\| I_{f_q}(q) - I_l(q) \right\| \right) < t_H \\ \infty & \text{else,} \end{cases} \tag{2.18}$$

where I_l is input image l. This essentially says that the values of pairs of pixels p and q, resulting from a transition in labeling between these two pixels, must match values found in at least one input image. This can be considered a much more strict version of the smoothness constraint, and works to avoid slicing of moving objects, which could be allowed in some cases by the smoothness term. The threshold should be quite strict, and is set to 5% in experiments by the authors.

The energy function is minimized using the expansion move algorithm provided by Boykov and Kolmogorov [60]. An application to High Dynamic Range Imaging is also demonstrated. For this applications, the authors convert the images to normalized radiance maps following the method of Robertson et al. [7]. Since the accuracy of radiance values are critical, exposure times for the normalization are adjusted to minimize the sum of squared differences of adjacent exposures. Under-/over-exposed pixels are excluded from the stationariness term, from the probability density estimation for the likelihood term, and from the approximated background model $\bar{I}$, using an HDR weighting function. The naive HDR result (see Section 2.2.1) makes a good choice for $\bar{I}$. Poorly-exposed pixels are assigned infinite costs to force them out of the solution. The resulting labeling is taken as an intermediate radiance map. This is used to composite a final HDR image using only values from the input images which agree well with it, similar to Gallo et al. [32].

Graph Cuts with Energy Function from Stationariness and Inpainted Values

Chen et al. [34] provide a similar overall method to that of Granados et al. Their data term is composed of a *stationary* term and a *predicted* term, where the stationary term for a given pixel and label is just the sum of absolute differences from all the other possible labelings for that pixel. The novelty in their proposal is in the predicted term. For this, they first determine which areas of the image sequence have been corrupted by movement. This is done by performing change detection between the first image and all the subsequent images. Any pixels in the sequence found to have changed by more than 20% with respect to the first image are marked as unstable.

Next, the first image is inpainted in the unstable regions; the inpainted values are assumed to be an indication of what *should* be in the unstable regions. For the rest of the images, the predicted data term is calculated in the unstable regions as the absolute difference of the pixel value in that image and the inpainted image. For the image inpainting, Chen et al. use a weighted average over a 101×101 pixel window around the current unstable pixel with weights inversely proportional to the distance from the

current pixel. The inpainting only considers stable pixels in the weighted average. If less than 50% of the pixels in this local window are stable, then the sampling window is extended to include the entire image.

The smoothness term is the same as Cohen's [58], except that the weighting factor γ is dropped. The energy is minimized using the expansion move algorithm implementation in the energy minimization framework presented by Szeliski et al. [61]. For an HDR application, the authors use the HDR sequence from [33]. It is converted to normalized radiance maps using methods from [2], which are then individually tonemapped and used as LDR inputs to the algorithm.

Several weaknesses with the approach outlined here present themselves. First, the weighting terms used by both Cohen et al. and Granados et al. are dropped, and the data and smoothness terms are not normalized, making the algorithm hard to tune overall. Second, the inpainting, being a simple weighted average, will tend towards a grayish colour throughout the inpainted region for most real world scenes. Third, since the inpainting defaults to a global weighted average, and this must be calculated separately for each pixel, the inpainting becomes very expensive when large portions of the scene are motion-corrupted since many pixels will then be inpainted globally. In one test on a 1024×768 image in which approximately 25% of the overall image was motion-corrupted, total run time was around 52 hours, most of which was taken up by the inpainting procedure. Fourth, the HDR application, while producing a final image with all the advantages of a tonemapped HDR image (all parts of the image appear to be exposed correctly), does not actually produce an HDR composite image at any point.

The deghosting methods covered in this section are summarized and compared in Table 2.1.

2.4 Segmentation Techniques

In Chapter 1, we proposed a revisiting and enhancement of change detection-based ghost removal, with the integration of over-segmentation techniques among the enhancements. Their integration into our system will be detailed in Chapter 4.

Many of the existing ghost removal techniques use some form of pixel grouping to refine the detection of motion-corrupted areas. This can be as simple as applying an averaging filter to images before detecting motion, and more commonly is done with morphological operators to bring together detached regions and eliminate small ones.

Since ghosts are typically salient objects, segmentation techniques offer a more pow-

Class	Method	Pros	Cons
Optical Flow	Global homography [28]	• Handles full frame-rate video	• Struggles with large displacements
	SR-HDR [29]	• Can compensate for large, complex camera displacements • Provides super-resolution enhancement	• Does not handle large, rapidly displacing ghosts • Computationally expensive
Change Detection	Variance and Entropy methods [30, 31]	• Simple and fast	• Replaces all motion-corrupted regions using single input image
	Extending dynamic range of reference image [32]	• Single reference image inherently deghosts	• Assumes one or more input images cover DR of scene sufficiently to act as reference
Background Modeling	Downweighting by Kernel Density Estimation [40]	• In best case, models background	• Assumes background seen more often than foreground • May be slow to converge
	Graph cuts with entropy-tuned energy [33]	• Can model background in very motion-cluttered scenes	• Very computationally expensive • Struggles with small/ distant ghosts
	Graph cuts with inpainting [34]	• In ideal scenarios, models background	• In testing, does not generalize well • Final result is LDR in current implementation

Table 2.1: Comparison of existing deghosting methods.

erful way to group them into parts along meaningful boundaries so that related groups of pixels can be analyzed together. Here we provide a brief review of Marker-based Watershed segmentation, a popular method for refining incomplete segmentation masks, as well as a class of over-segmentation known as superpixels.

2.4.1 Marker-based Watershed Segmentation

Conceptually, the Watershed transform views an image as a topographical map where bright pixels represent higher altitudes and dark pixels lower altitudes [62, Ch.10]. Lower regions in a local neighbourhood (i.e. local minima) are known as catchment basins while higher ridges separating them are called ridge lines. The Watershed transform finds the ridge lines and catchment basins of an image. So, the strategy for using the Watershed for segmentation is to figure out how to "change your image into another image whose catchment basins are the objects you want to identify" [63].

Various approaches to calculating the Watershed transform have been proposed [64–66], and many additional methods have been proposed to use the Watershed transform for segmentation; see Gonzalez et al. [67] for details. The rest of our discussion on Watersheds is based on Gonzalez et al. [62, Ch.10] and a MATLAB product demo [68]. Readers looking for further details on practical Watershed segmentation should refer to tutorials by Eddins [63, 69, 70].

A common starting point for Watershed segmentation is to use the gradient of an image as the input to the Watershed [68]. Since the gradient has high values along object edges and low values elsewhere, the watershed ridge lines should follow the edges nicely. However, most images will have many small local variations in intensity in addition to the major object edges, and the result is over-segmentation. Over-segmentation can be solved by reducing local minima across the image so that the only minima that exist are the regions you want segmented or separated from on another. This means calculating a set of foreground and background markers. Once these have been found, they are used in a minima imposition algorithm [71, Ch.6] to modify the gradient image before calculating its watershed.

The result is a series of ridge lines (image value of 0) dividing the regions that were marked, and numbered catchment basins elsewhere. Table 2.2 shows some basic Watershed segmentations, and in Section 4.2, we explore the integration of marker-based Watershed segmentation into our proposed deghosting method.

2.4.2 Superpixels

Superpixels provide an intentional over-segmentation of an image which can become an input to a variety of algorithms in which we want to refer to meaningful local groupings of pixels, i.e. groupings respecting salient boundaries.

Normalized Cuts

The term Superpixels was introduced by Ren and Malik [72], who make a concise case for their need. They point out that 1) even at moderate resolutions, the number of pixels in an image make many algorithms intractable when executed at a per-pixel level, necessitating a method for referring to groups of pixel, and 2) superpixels as outlined in their work reduce the number of pixels of even high resolution images to just a few hundred local, compact, and most importantly, coherent segments. Their work focuses on using a two-class model for learning segment grouping. Their classes are drawn from ground-truth segmentation by humans, which are used directly as positive examples of correct grouping of pixels, while negative examples are created by randomly pairing a ground truth segmentation from one image with another image. They introduce superpixels as a pre-processing step before grouping them to form major segments of the image. Their approach builds on earlier work on Normalized Cuts (N-Cuts) [73, 74]; here they use contour and texture cues.

Mori [75] adopts the superpixels of Ren and Malik in a human pose recognition algorithm. One of the steps of the algorithm is to identify human half-limbs (e.g. forearms), and superpixels are used to reduce the search space from the per-pixel level to just a few hundred superpixels. Mori has released MATLAB code for N-Cuts superpixels.

Segment Growth by Geometric Flows

Levenshtein et al. [76] aim to improve existing superpixel work with "Turbopixels." They point out that the Watershed, when used for over-segmentation, lacks a compactness constraint. This leads to under-segmentation when boundary (edge) information is lacking or is of very low-contrast. This can be solved by the introduction of a compactness constraint, similar to N-Cuts. However, N-Cuts uses a global optimization which is computationally expensive. While Shi and Malik were able to reduce the complexity to $\mathcal{O}(N^{\frac{3}{2}})$ where N is the number of pixels, and later methods offered a further reduction by $\sqrt{N}$, these came at the cost of no longer controlling the number of superpixels desired, as well a loss of uniformity in shape and size.

The Turbopixels method instead grows a series of K segments (superpixels) using constrained geometric flows. The constraints used are:

- *Uniformity* Segments should be approximately of uniform size and shape, with small deformations to suit local features like edges.
- *Connectivity* All pixels within a segment should have a high degree of connectivity, i.e. they should all be related in a meaningful way.
- *Compactness* In the absence of local edge information, segments should tend toward compactness to prevent runaway behaviour.
- *Smooth Edges* Superpixel edges should follow image edges.
- *No Overlap* Superpixels should not overlap, i.e. all pixels in the original image should fall into one single superpixel.

The authors have released MATLAB code with pre-compiled Mex binaries.

Simple Linear Iterative Clustering

Achanta et al. introduce Simple Linear Iterative Clustering (SLIC) Superpixels [77], a modification of k-means clustering. Here, clustering is performed in 5D space defined by the three coordinates of the $L^*a^*b^*$ (CIELAB) colour space and two x, y spatial coordinates. They introduce a new distance measure to normalize measurements in Lab and xy with respect to each other for clustering purposes. The CIELAB colour space is used due to its approximate perceptual uniformity. Like [76], they enforce compactness to achieve a segmentation in which all superpixels are of similar area and shape. They provide an excellent review of existing over-segmentation approaches including several not covered here. Finally, they show their approach to be faster than previous state of the art approaches while providing similar or better results in both common test sets and real-world applications.

A brief outline of the SLIC algorithm is as follows. First, K seeds are placed on a grid pattern, where K is the number of superpixels desired. For an image with N pixels, this means an approximate area of N/K per superpixel, and a grid spacing of approximately $S = \sqrt{N/K}$. Since the approximate size of each superpixel is $S \times S$, they restrict their search for associated pixels to within a $2S \times 2S$ neighbourhood around the current center. Since the Lab colour coordinates are limited in size, but the x, y space coordinates

are not necessarily, they normalize them through a special distance measurement, D_s, that replaces the common Euclidean distance. This is done by normalizing the distance measure in each space for pixels i and j as follows:

$$D_s = d_{lab} + \frac{m}{S} d_{xy}, \tag{2.19}$$

$$d_{lab} = \sqrt{(l_i - l_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2}, \tag{2.20}$$

$$d_{xy} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \tag{2.21}$$

where m serves as a compactness factor. Increasing it in a recommended range of $[1, 20]$ results in more compact superpixels, possibly at the expense of edge preservation. The authors use a value of 10, which approximates the maximum perceptually meaningful distance in the CIELAB colour space and was found to provide a good balance of the colour similarity and spatial proximity constraints.

After the K seeds are placed, they are moved to the minimum gradient position in their 3×3 neighbourhood to avoid starting them on an edge or a noisy pixel. Each pixel is then associated with the nearest seed or "cluster center" whose $2S \times 2S$ neighbourhood overlaps it. Next, an updated center is calculated as the average $labxy$ vector of the pixels associated with that center. Then pixels are re-associated with the nearest cluster centers, and this process iterates until convergence. After convergence, there may be some stray pixels that are not associated with a cluster; a final step associates any stray pixels with the nearest cluster center. The authors have released C++ source code.

Since this is a special case of k-means clustering, the authors provide a comparison with standard k-means, which is $\mathcal{O}(NKI)$ for N image pixels, K superpixels, and I iterations. Since SLIC reduces the search area to the $2S \times 2S$ neighbourhood, and this area is in fact inversely proportional to the number of superpixels requested, they are are able to achieve $\mathcal{O}(N)$, meaning linearity in the number of image pixels regardless of the number of superpixels requested.

The authors provide extensive comparisons with existing state of the art methods, including two applied examples. On the Berkeley segmentation test set, they find their algorithm to be 10 times faster than Turbopixels and 500 times faster than N-Cuts. In an object recognition test, they show superior performance of SLIC, and notably, in a medical image segmentation test, they show slightly superior quality segmentation to

N-Cuts and vastly superior to Turbopixels which, due the absence of strong gradients in the image, merges many pixels resulting in an under-segmentation.

Comparison of Superpixel Segmentation Techniques

Table 2.2 compares the strengths and weaknesses of three superpixel segmentation methods, with the well-established Watershed as a reference pseudo-superpixel segmentation. Figure 2.12 demonstrates the results of the same methods. All superpixel results have approximately 200 segments or superpixels, and were performed on a 640x480 source image. The Watershed segmentation was performed using a method from [62] in which the gradient of the grayscale image is smoothed with morphological operators to remove local minima before calculating the Watershed. Smoothing with a larger structuring element reduces the number of segments, giving a rough control over the number of segments. In the first example, notice that the Watershed follows strong edges but does not respect the fuzzy edges along the occlusion boundary of the dog. As discussed, we see that the SLIC method provides the best combination of speed, edge following, and compactness constraint in the absence of strong edges.

The N-Cuts result shows a marked improvement over the Watershed, though compactness is lost in some of the darker regions, manifesting as long drawn out regions. This is the most computationally expensive algorithm, and took approximately 6 minutes to complete. In tests on a 1024x768 image, the resolution of the Ladybug2 sensors, it failed to complete due to memory issues, and was therefore deemed unsuitable for our application. Turbopixels completed in 26 seconds, but the quality of edge following seems lower compared to N-Cuts. Finally, the SLIC algorithm completed in less than a second with edge following comparable to N-Cuts and finer detail tracing in places. It also produced segments with greater compactness; notice that in large uniform regions, the pixels appear grid-like due to the lack of edge information, which engages the compactness constraints. Further discussion and examples of the integration of marker-based segmentation techniques into our system are provided in Section 4.2.

2.5 Tonemapping

In the context of HDR imaging, *tonemapping* refers to a class of algorithms designed to compress the large dynamic range of HDR images, often 6 orders of magnitude or greater, into a range that can be displayed on standard monitors, which are typically

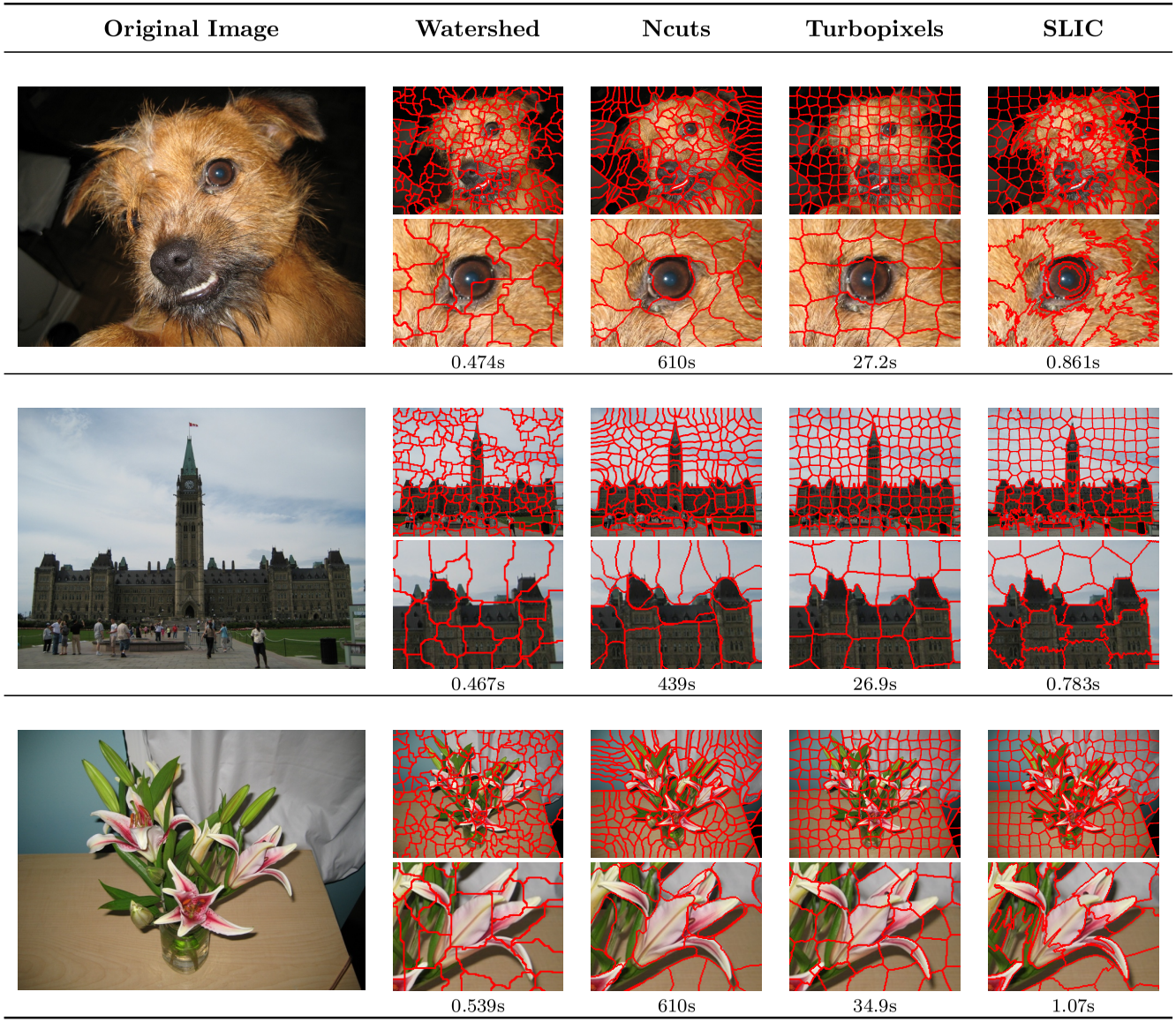


Figure 2.12: Examples of superpixel segmentation methods. Full size results on top of results rows, and magnifications on bottom. All segmentations have approximately 200 regions. Run times are averaged across 10 executions on the same data set.

Method	**Pros**	**Cons**
Watershed	• Fast • Follows strong edges	• No compactness constraint; runaway behaviour in abscence of image edges • No explicit control over number of regions
N-Cuts	• Global optimization • Better edge following than Watershed	• Very slow • Less compact regions than other methods • Prohibitive memory constraints
Turbopixels	• Much faster than N-Cuts • Compact superpixels	• Edge following not as good as N-Cuts in testing
SLIC	• Fastest method by far (nearly real-time) • Excellent edge following • Compact superpixels • Explicit control of number of regions	• Some pixels may take on ungainly shapes, e.g. near-toroidal crescents (see Figure 2.12)

Table 2.2: Comparison of Superpixel segmentation methods. Watershed is used as a pseudo-superpixel method as in [62].

limited to around 500:1 at best for a static image [30, Ch.1]. A full review of tonemapping theory and algorithms is outside the scope of this thesis so we provide a brief overview and some examples of popular and/or freely available tonemappers, and refer readers to several existing tonemapper reviews for more detail [30, 56, 78]

Tonemappers are often broken down into global and local operators, although some operators use a combined global/local scheme. Global operators apply the same compressive function to every pixel; the function may change image to image based on global statistics such as mean value, but within an image it does not change. Local tonemappers use compressive functions that depend on information in the pixel's local neighbourhood. For example, high contrast regions may undergo quite different compression than low contrast. Another common distinction is photorealistic versus surreal operators. Photorealistic operators attempt to create an image that looks as much like the original scene as possible while still giving the impression of containing an expanded

dynamic range. Surreal operators tend to emphasize metrics like local contrast, so that in local neighbourhoods, details are well-preserved, but over the entire image, the effects can be quite unnatural as one portion of the scene which should clearly be darker than another can end up lighter. These operators, often used for artistic effect, are responsible for the so-called "HDR look."

The simplest tonemappers are global algorithms like linear or logarithmic compression. Linear compression compresses the HDR by dividing by the maximum value and rescaling into an 8-bit range. This is fast but tends to make large sacrifices in details (contrast). If the scene contains very bright values, all values above a chosen percentile can be clamped to retrieve details in dark areas, but this results in saturation in the bright areas. Logarithmic compression gives a marked improvement in visible detail, and provides a good baseline for global algorithms. However, it tends to give a "washed out" look, as seen in Figure 2.13a, where the base-10 logarithm of pixel values was rescaled and quantized into an 8-bit range.

More advanced global tonemappers have been proposed [79–85]. Larson et al. [82] use models of human sensitivity to contrast, glare, and other visual cues to guide a histogram adjustment process. A modified version of this algorithm is used in MATLAB's tonemapping function, and the results are shown in Figure 2.13b. Pattanaik et al. [85] propose a global tonemapping method, shown in Figure 2.13c, based on an understanding of how our eyes adapt to available luminance over time. It was orginally developed for HDR video, but has since been adapted for still images, and a local version was published later [86]. Drago et al. [79] use logarithmic compression but adaptively vary the logarithmic base to preserve details; an example is shown in Figure 2.13d. Reinhard and Devlin [84] propose a global tonemapping function based on a computational model of photoreceptor sensitivity. Their algorithm has four user-controlled parameters, as well as a mechanism to provide estimates of these parameters when a fully automated process is necessary. Notably, one of these parameters balances interpolation between local and global pixel intensity, which is an input to their compressive function, with the result being that their algorithm can also be used as a local tonemapper. A sample is shown in Figure 2.13e.

While global tonemappers tend to be computationally efficient, they often sacrifice details. By relying upon information within a pixel's neighbourhood, local tonemappers can often achieve greater compression and retain greater contrast levels, and hence give the appearance of greater detail retention within some portions of the image [87–96]. However, this often comes at a cost of increased halo effects around sharp edges, as well

as contrast reversals wherein a portion of the scene which was darker than another in the original scene ends up brighter in the tonemapped image.

Durand and Dorsey [89] decompose the HDR image into a "base" and "detail" images, essentially low pass and high pass filtered images. The base layer is then scaled to fit into the target range, and the detail layer is multiplied back into the compressed base to produce the tonemapped image. The base is obtained using the edge-preserving bilateral filter, and a fast version of the filter is introduced. Fattal et al. [94] perform local tonemapping in the gradient domain by spatially varying how gradients are attenuated, specifically attenuating small gradients less than large ones. Reinhard et al. [91] use a sigmoidal compressive function, which has been found to model the response of photoreceptors in the human eye. In its default form, it is a local operator, but it offers a parameter which can be used to balance the contribution of global/ local information, allowing it to be turned into a global operator.

Mantiuk et al. perform tonemapping in a "contrast domain," leveraging the fact that the human eye is only able to distinguish fine details or high frequencies over about 1.7 degrees of the visible field [97]. Contrast can therefore be emphasized over small neighbourhoods to increase detail retention. They introduce two versions of their method, contrast mapping and contrast equalization, with the former providing a more photorealistic result, and the latter tending to emphasize local contrast most heavily. Mantiuk et al. [98] introduce a method for tonemapping for specific combinations of display device and ambient lighting such as e-paper, lcd screens, flourescent lighting, sunlight, etc., factoring in how the human visual system adapts to these scenarios. The *pfstmo* implementation[3] offers various preconfigured display/lighting combinations such as LCD-Office for an LCD display viewed in a typically lit office environment [99]. Recently, Paris et al. [100] used Laplacian pyramids to perform a variety of edge-aware image processing tasks, including tonemapping and inverse tonemapping.

A selection of the tonemappers reviewed here are demonstrated in Figure 2.13. Result a was obtained manually in MATLAB, b uses MATLAB's *tonemap* function, l uses the authors' code from the associated website [101], and the rest were obtained using the *pfstmo* package [99].

[3] http://pfstools.sourceforge.net/pfstmo.html

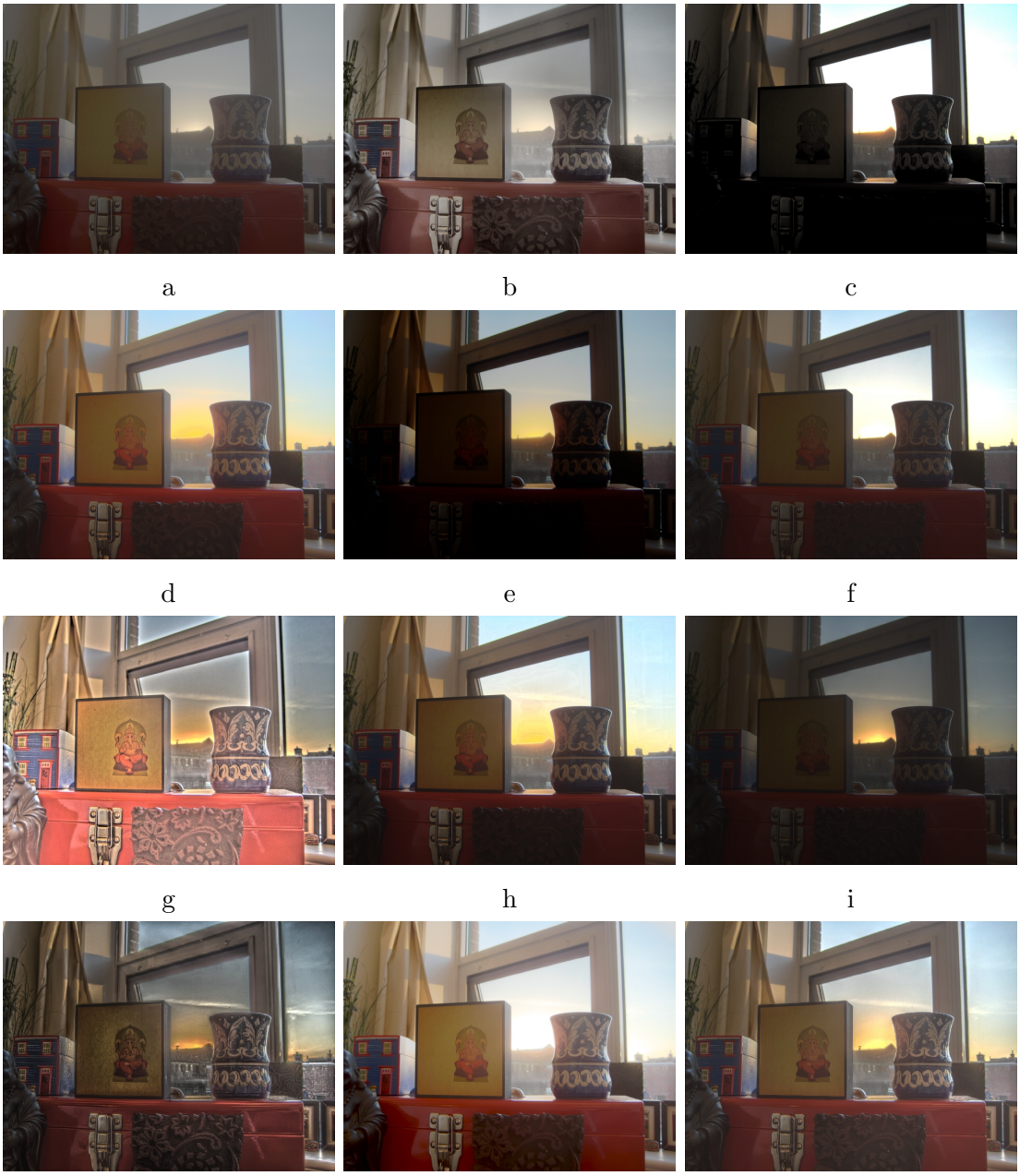


Figure 2.13: Comparison of tonemapping operators. a Logarithmic compression. b Histogram adjustment [82]. c Time-dependent adaptation [85]. d Adaptive logarithmic [79]. e Photoreceptor physiology [84]. f Fast bilateral filtering [89]. g Gradient domain compression [94]. h Photographic tone reproduction [91]. i Local contrast mapping [97]. j Contrast equalization [97]. k Display adaptive [98]. l Local Laplacian filters [100].

Chapter 3

Capture Setup

We capture images using the Point Grey Research Ladybug2 ominidirectional camera, shown in Figure 1.1. The manufacturer offers two HDR capture modes, which are outlined in the following section. Both of these methods have proven too limited for our purposes so we have developed a custom exposure-bracketed capture mode for the Ladybug2. After capture of exposure-bracketed RAW images and associated exposure data, we can apply a naive HDR approach for each CCD, with fixes for degenerate values. These HDR CCD views can then be remapped into a chosen panoramic projection for later display in a panorama viewer. These steps are detailed in this chapter, while deghosting methods necessary for non-static scenes are detailed in Chapters 4 and 5.

3.1 Ladybug2 HDR Modes

The Ladybug2 comprises six 8-bit, 1024x768 sensors in a single housing, each with a wide angle lens, fixed aperture, and Bayer-pattern Colour Filter Array. This configuration allows simultaneous capture of about 75% of an entire sphere, but provides limited dynamic range over any single sensor. Two options exist for capturing HDR, though a complete workflow has only recently been added to the camera's SDK for the first option, and the second option is not fully implemented in the SDK.

In the first option, the Ladybug2 allows all sensors to be set to auto-expose, meaning that in one shot the camera can capture a greater dynamic range over the full spherical view as compared to what can be captured by a single sensor. While this can provide dynamic ranges of four orders of magnitude [30], there are several limitations to this approach. Notably, in very high dynamic range scenes, it is easy for two adjacent sensors

to capture portions of the scene with drastically different intensities, using very different exposure settings. When the captured images are normalized, this will tend to amplify dark regions and make them appear very noisy. A second limitation is that the sensors capture 8-bit RAW data, creating an upper limit on dynamic range of 255:1, which in practice will be lowered to as little as 50:1 by noise effects, so the dynamic range of individual sensors are easily exceeded by real-world scenes.

Another option is the camera's "hardware HDR mode." When this mode is used, four registers in the camera store four combinations of gain and exposure time. The camera then cycles through these at frame rate, and outputs the exposure-bracketed images. This mode again has several limitations. First, for any given bracket, the same exposure settings are applied to all 6 CCDs. With only four possible settings, we again have trouble covering many real-world scenes. For example if one of the side sensors is looking at shadows while the top sensor looks at a brightly lit sky, it can be difficult to cover this complete range with just four settings. Second, the HDR mode itself has some bugs which were discovered in early testing. These relate to the camera's video camera style output and lack of synchronized meta data; the camera is continually capturing and outputting frames at the current frame rate. However, when capturing RAW data, these frames do not include meta data, so exposure settings need to be read separately, and there is no reliable way to synchronize the two when capturing at full frame rate.

Due to the limitations of these two HDR modes, a custom exposure-bracketing mode for the Ladybug2 has been developed, and is outlined next.

3.2 Capture

We have developed a custom capture program based on an existing utility for an LDR panorama capture pipeline, and using the same hardware configuration [23–25]. In the existing capture setup, a Ladybug2 is mounted on a mobility scooter, along with a GPS unit and an inertial sensor. All hardware is connected to a laptop and powered with a portable battery pack. Point Grey Research provides a Software Development Kit (SDK) consisting of several C++ libraries providing access to camera hardware settings, as well as data capture and processing functionality, some of which we have used in our work.

Our capture process consists of first setting the Ladybug2 so each sensor is in an independent auto-exposure mode and therefore chooses the best exposure for the portion of the scene it currently views. When a capture event is triggered, either manually or by an interval timer, the sensor's auto-exposure value becomes the reference point for

an exposure-bracketed image sequence. Additional exposures are chosen using geometric spacing in fractions of Exposure Value (EV), where we have chosen EV scale spacing to conform to the existing HDR and photography literature.

A new exposure is calculated by multiplying the current exposure by 2 raised to the power of the negative of the EV change. For example, if a camera's auto-exposed setting gives an exposure time of $\frac{1}{30}$ and we are using an EV spacing of 1, the +1 EV exposure, calculated as $\frac{1}{30} \times 2^{-1} = \frac{1}{60}$, would be half as bright and the -1EV exposure, $\frac{1}{15}$, would be twice as bright. Note that a decrease in EV corresponds to an increase in brightness, and an EV change of 1 corresponds to an exposure factor of 2. In HDR imaging, the reference exposure is usually referred to as having an exposure of 0 EV. Some confusion can arise since, in practice, HDR image capture is often done using the *exposure compensation* settings of cameras, which are opposite to the EV scale so that an exposure compensation of +1 actually doubles brightness. So, when we say increasing *exposure*, we mean brighter images, while increasing *EV* would mean darker images. We will mostly speak in term of exposure as it is more intuitive. During the capture process, we capture RAW image data to avoid compression-related quality loss associated with the camera's optional JPEG mode. RAW data is captured to disk along with exposure settings used, and all other processing is done offline after capture is complete.

We offer two simple modes for users to choose from, the first being mostly manual, and the second being mostly automatic.

In the first mode, the capture program allows the user to choose an EV spacing and number of bracketed images at startup, and each capture event consists of capturing that number of bracketed images for each of the camera's CCDs. In practice, the exposure time is set to a digitial register which may not allow a setting corresponding exactly to the desired speed. However, the camera internally converts requested values to the nearest allowed register setting, so after requesting a shutter speed, we read it back to get the actual value used. Exposure can also be manipulated via the gain setting, but this increases noise in the image. Currently, exposure settings exceeding the min and max values allowed by the camera are simply clamped to the min/ max allowed. Within this mode, several common bracketing orders have been implemented, including $\pm$ in which exposures alternate between brighter and darker in increasing EV steps, $+$ in which the exposure is always increased from the reference image, and $-$ in which it always decreases.

We also add a modified version of $\pm$ bracketing in which exposure settings are chosen in a $\pm$ manner, but rearranged into ascending exposure before capture, with the result

being that in time-sequential order, the reference image is in the middle, and the portion of correctly exposed pixels shared by adjacent images is maximized. For example, if the user has chosen to capture a set of 5 images at 2 EV spacing, for a CCD with a current auto-exposed shutter time of $\frac{1}{240}$, the complete bracketed sequence would be $\{+4, +2, 0, -2, -4\}\,\mathrm{EV} = \left\{\frac{1}{3840}, \frac{1}{960}, \frac{1}{240}, \frac{1}{30}, \frac{1}{15}\right\}$. The advantages of this bracketing order will become more obvious in Chapter 5; briefly, it allows comparison of a greater fraction of adjacent images which is advantageous for ghost removal, and will likely be advantageous in any future work on image alignment. In the manual mode, we choose exposure-bracketing settings (numer of images, EV spacing) manually to tailor them to the capture scenario. This may be challenging for new users, hence our addition of an automatic mode.

In the automatic capture mode, the user chooses only the desired EV spacing. Then, during a capture event, the capture program decreases exposure without saving images until either all pixel values across all CCDs are under 220, or all CCDs' shutter times are at the minimum allowed setting. Once one of these conditions is met, we begin alternately capturing and increasing exposure until either the max exposure time is met for all CCDs, or the minimum value is 20. These value limits are based on recommendations in [30, Ch.4]. The advantage of this mode is that it is simple to configure, it only captures the number of images necessary to capture the range of the scene, and it is no longer forced to capture a symmetric number of images about the auto-exposure setting. In a given real world scene, it might only be necessary to capture 2 images brighter than the auto-exposed images, but 5 darker. This is not possible with the manual mode, but is done automatically by this mode. The user can vary the number of images required to capture the range by varying the EV step, and if it is determined in the future that this mode is still capturing too many images in its attempt to reach the proposed limits, they could be relaxed somewhat. For example, it may be sufficient to only require that 5% of pixels be under 220.

3.3 White Balancing

In [24], white balancing is performed purely in software using the grey world assumption. In grey world white balancing algorithms, it is assumed that the average colour the camera has seen is grey, meaning the averages of all colour channels should be the same. If the averages are not equal, then two of the colour channels are scaled in reference to the third to make their averages equal. While the assumption can be violated in

some situations, it is quick and accurate enough for most scenarios to be acceptable. In a panoramic imaging scenario, the assumption is made stronger by sampling over the entire panorama so that more of the scene is included, and even over multiple panoramas as in [24].

The Ladybug2 offers a hardware/ software Auto White Balance (AWB) mode in which the camera is white balanced by varying two gain registers which scale the red and blue colour channels in reference to the green channel directly on the CCD so that the RAW images are white balanced. Not using this feature runs the risk that we may not use the full range of one or more of the channels, so we have added this functionality to our capture pipeline. A user can now trigger an AWB event during capture. The algorithm terminates after either a good balance is achieved or a maximum number of iterations is reached.

The SDK provides the function *ladybugDoOneShotAutoWhiteBalance* that begins the AWB process. After a call to *ladybugDoOneShotAutoWhiteBalance*, subsequent calls to *ladybugConvertToMultipleBGRU32* (the SDK's demosaicking function) will make adjustments to the white balance registers. In practice, the algorithm will often take a long time to converge or exceed the maximum allowed number of iterations, but the result is still usually visually pleasing. Therefore, the user is notified that the AWB is underway and that they can wait for it to complete or simply commence capture once the on-screen images, which are updated in real time as the AWB converges, appear to have reached a good white balance point. In our experiments, the images typically appear to be well white balanced after just a few frames (1-2 seconds).

3.4 Naive HDR Combination

The process of naive HDR combination, which assumes no motion of the camera or scene elements, is introduced in Section 2.2.1. To review briefly, we first capture a series of exposure-bracketed images of a scene. Samples of these images are then used to perform radiometric calibration, recovering the camera's photometric response which is typically non-linear and unknown. Once the response is known, it is used to convert the images to irradiance values in which pixel values within an image are linearly related to scene irradiance. From here, they are normalized by dividing by exposure time and/ or linear gain. Then, the images are combined using a weighted average, where the weighting function is chosen to downweight pixels whose values are considered less reliable, typically those towards the extremes of the sensor's range. The result is an HDR image.

3.4.1 Photometric Calibration Methods

Here, we expand on the brief review of naive HDR literature provided in Section 2.2.1, before discussing naive HDR combination in the context of our system.

Method of Debevec and Malik

Debevec and Malik [2] introduced a generalized method for HDR recovery, laying the foundation for modern HDR recovery. Two critical contributions of their work are: 1) a generalized method for recovering a camera's response curve which allows us to convert LDR image values into more physically meaningful linear radiance values, and 2) a method for taking a weighted sum of these linear values to compose a final HDR image.

The reason for needing to recover a camera response curve is that, as shown in Figure 2.2, a camera's overall response to real-world radiance values involves multiple stages, one or more of which may be non-linear, so that the overall response is unknown. This was a major issue with film cameras, and although the linear response of CCDs over most of their operating range [5, Ch.5] makes the overall response more linear, conversion to common LDR storage formats such as JPEG typically involves some non-linear mapping stage such as a gamma curve. The camera's response curve, which maps digital pixel values, Z, to linear irradiance values, E, at the sensor, can be recovered from a series of differently exposed images if reciprocity holds. Reciprocity is the concept that only the product, $E\Delta t$, of E and the exposure time, Δt, is important to the exposure at the sensor, $X = E\Delta t$, so that a halving of E combined with a doubling of Δt will result in the same value for X [2]. Reciprocity typically holds for the entire operating range of most cameras.

Assuming that E is constant over the different exposures for a given image, and that reciprocity holds, then Z is related to it as follows, where i indexes over pixels in a given image and n indexes over different exposures:

$$Z_{i,n} = f\left(E_i \Delta t_n\right) \tag{3.1}$$

The authors further assume that f is monotonically increasing and therefore invertible so that

$$f^{-1}\left(Z_{i,n}\right) = E_i \Delta t_n \tag{3.2}$$

Taking the logarithm of both sides and defining a new function $g = \ln f^{-1}$ gives

$$g\left(Z_{i,n}\right) = \ln E_i + \ln \Delta t_n \tag{3.3}$$

A collection of differently exposed images therefore produces a set of equations sampling segments of the the currently unknown function g. The authors develop an objective function and minimize it via singular value decomposition to render g in the form of a lookup table; the minimization takes advantage of the fact that there are a finite and in fact small number of values Z can take on, for example 256 in the case of 8-bit images. Once we have g, it can be used to linearize the pixel values of the LDR image so that they are then only scaled by their exposure time. The images can then be normalized in exposure space by dividing the pixel values of each image by its respective exposure time, Δt_n.

At this point, we theoretically have a series of HDR images of the scene and if the sensor used to capture them did not suffer from under-/ over-saturation effects, they would all be the same image. In reality, the saturation behavior of CCDs means that pixels values very close to the extremes of the sensor's range do not tell us anything. This can be easily visualized: picture a pixel with an 8-bit ADC exposed to a variable light source. As the light source intensity increases, the pixel's RAW value increases linearly. However, once it hits the maximum value of 255, it stops increasing even though the light source continues to get brighter. As such, an infinite number of irradiance values above the first value which saturates the pixel will all map to the same digital pixel value of 255. The situation is similar for the minimum pixel value. Furthermore, because of noise effects in CCDs, pixel values near but not actually at the maximum are still unreliable. So, pixels in these regions in the LDR images should contribute less to the final HDR value than pixel values near the center of the range of the CCD. This can be easily accomplished with a weighting function that down-weights pixels near the extremes. Debevec and Malik propose a triangle hat function defined as

$$w(z) = \begin{cases} z - Z_{min} & \text{for } z \leq \frac{1}{2}\left(Z_{min} + Z_{max}\right) \\ Z_{max} - z & \text{for } z > \frac{1}{2}\left(Z_{min} + Z_{max}\right) \end{cases} \tag{3.4}$$

This weighting function is also used in the objective function to encode the fact that g will tend to be steeper near the limits of z and will fit the data more poorly in these areas. Now that we have a weighting function and images that are linearly related to each other and to the original scene radiance, it is trivial to combine them to create the final image, H:

$$H_i = \frac{\sum_{n=1}^{N} w\left(Z_{i,n}\right)\left(g\left(Z_{i,n}\right) - \ln \Delta t_n\right)}{\sum_{n=1}^{N} w\left(Z_{i,n}\right)} \tag{3.5}$$

Method of Mitsunaga and Nayar

Mitsunaga and Nayar present an alternate method [6]. They point out that Debevec and Malik's method is suitable when noise is minimal and the exposure times are precisely known. They then propose a new method that does not have these requirements and which uses a parametric model for the camera response function. They argue that since camera response curves are reliably monotonic or at least semi-monotonic, they can be approximated with a high order polynomial:

$$X = g\left(Z\right) = \sum_{m=0}^{M} c_m Z^m, \tag{3.6}$$

where m is an integer power and M is the order of the polynomial. The recovery of the camera response therefore consists of recovering both the number of coefficients, M, and the actual coefficients, c_m that best fit the measured data. Now if two images of the same scene are taken with two exposures e_n and e_{n+1}, letting $R_{n,n+1} = \frac{e_n}{e_{n+1}}$ and making use of expression 3.6, we get:

$$\frac{\sum_{m=0}^{M} c_m Z_{i,n}^m}{\sum_{m=0}^{M} c_m Z_{i,n+1}^m} = R_{n,n+1}. \tag{3.7}$$

The Sum of Squared Errors (SSE) of expression 3.7 is taken as the error function to be minimized. Expression 3.7 has obvious ambiguities since both sides can be raised to an exponent u so that it has multiple possible solutions for different u. However, the authors claim that the use of a polynomial model means that the solutions are well spaced so the system can still be solved if the search is restricted to the vicinity of one particular solution. The exposure settings supplied by the user, which in turn give R, provide a good initial estimate. When there are more than two images, the computational load is reduced by using an iterative solution in which the ratio estimates at the end of each iteration are taken as the new initial estimates for the next iteration. Their objective function is minimized for a given polynomial order M, and this is repeated for orders up to an arbitrary maximum $M_{\max}$ (the authors use 10) to find the M that gives the minimum SSE.

For a weighting function, the authors argue that measurements of radiance are most reliable when their signal to noise ratio (SNR) are at a maximum, which allows them to develop a weighting function as the ratio of the recovered response to its derivative:

$$w\left(Z\right) = \frac{g\left(Z\right)}{g'\left(Z\right)} \tag{3.8}$$

Once the response function has been recovered and the weighting function has been calculated, assembly of the HDR image proceeds as before. The strengths claimed by the authors are that their method uses a parametric model and is therefore more robust in the presence of noise, which is backed up by testing results from noisy synthetic data.

Method of Robertson et al.

A final method for recovering camera response is that of Robertson et al., who point out that for a given pixel exposed to a constant irradiance, quantization error is less significant for higher exposure times. Therefore, while a weighting function should give higher weight to pixels in the middle of the camera's digital output range Z, it should additionally give higher weight to pixels above the middle than below the middle. They introduce a new Gaussian-like weighting function,

$$w_{i,n} = w\left(z_{i,n}\right) = exp\left(-4 \cdot \frac{\left(z_{i,n} - Z_{mid}\right)^2}{Z_{mid}^2}\right), \tag{3.9}$$

where $Z_{mid} = \left(Z_{max} - Z_{min}\right)/2 = 127.5$ for 8-bit images, and $w(z)$ is re-scaled so that $w(Z_{min}) = w(Z_{max}) = 0$ and $w(Z_{mid}) = 1$.

The HDR combination function for a known response function $g(\cdot)$ is given by

$$E_i = \frac{\sum_n w_{i,n} t_n g\left(z_{i,n}\right)}{\sum_n w_{i,n} t_n^2}, \tag{3.10}$$

As such, the normalization in exposure space and weighting of the normalized values happen in one step, and the exposure times serve to weight values of z above Z_{mid} higher than those below. When $g(\cdot)$ is unknown, it must be estimated. Like Debevec and Malik, the authors assume a simple lookup table. They introduce an objective function as:

$$\tilde{O}(\mathbf{X}, \mathbf{E}) = \sum_{i,n} w_{i,n}\left(g\left(z_{i,n}\right) - t_n E_j\right), \tag{3.11}$$

Since this requires simultaneous estimation of the unknown irradiances E along with $g(\cdot)$, an addition constraint is introduced by stipulating that $g(Z_{mid}) = g(128) = 1.0$. This

means that the response function is only recovered up to a scale, so that the resulting HDR image has linearly related relative Irradiance. To produce absolute values requires performing absolute calibration with a known illuminant. The objective function is minimized using iterative Gauss-Seidel relaxation. This method is implemented in the the open source package *pfscalibration*[1].

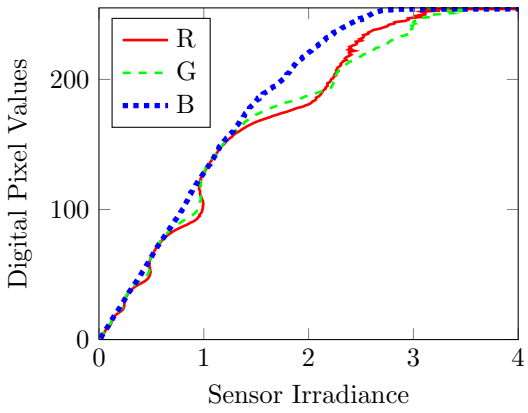
3.4.2 Photometric Calibration for Proposed System

If the camera response to irradiance is linear, the calibration step is unnecessary. CCD response to irradiance is typically close to linear over most of the range [5, Ch.5], so when creating HDR images directly from RAW images calibration would be unnecessary. However, because one of our goals is ghost detection, which is much easier to perform when colour information is available, we perform demosaicking prior to applying the naive HDR process. We use a demosaicking function from the Ladybug SDK, and early calibration tests were inclonclusive as to the linearity of the demosaicking algorithm, so we further investigated the calibration of the camera. Ultimately, we found that even with demosaicking, the response is close enough to linear over enough of the sensor's range that we can safely assume a completely linear response. Our experiments are detailed here.

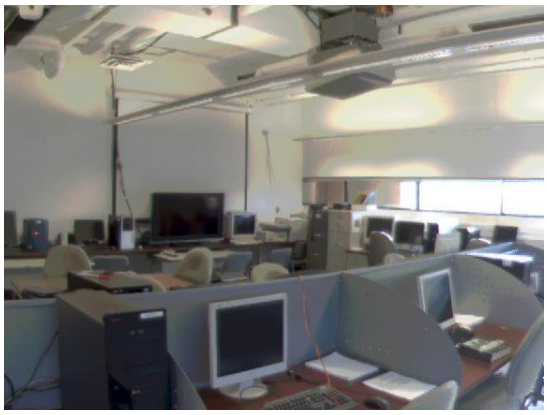
For calibration experiments, we used the method of Robertson et al. as implemented in *pfscalibration*. Initial testing revealed that Robertson's method can produce poor calibration results in many situations, notably in those with bright highlights. This means that it is not possible to use self-calibration, where a bracketed set of images are used to produce both the camera response curve and the HDR image rather than using a previously determined calibration curve. Figure 3.1 illustrates a poor self-calibration scenario. The scene contains bright highlights and mixed lighting sources. The resulting curve shows non-monotonic behaviour in the form of switch-backs, which goes against the fundamental assumptions about camera response discussed in Section 2.2.1. The resulting HDR image, which is shown after tonemapping, displays banding in regions that should be smooth gradients, such as on the white walls.

However, we were able to obtain good calibration curves in some scenarios with consistent light sources and milder highlights. An example is shown in Figure 3.2 where we have applied a curve recovered from another CCD and scene to the same image set as Figure 3.1. We see that the response is now monotonic and the resulting image is

[1] http://pfstools.sourceforge.net/pfscalibration.html



(a)



(b)

Figure 3.1: An example of poor calibration results. a Robertson curve. Note switchbacks (non-monotonic behaviour). b Portion of a tonemapped result. Note the banding and chromatic artifacts on the walls in transition from brightly lit to darker portions of the wall.

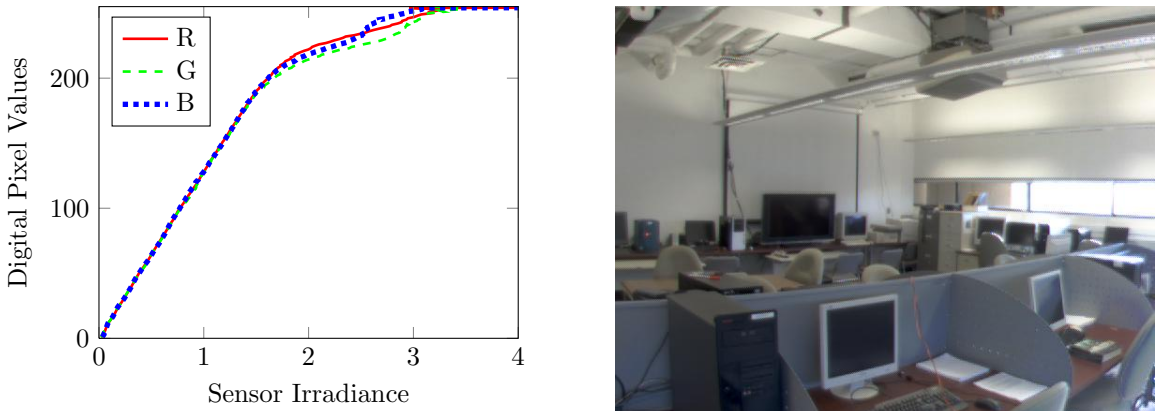
greatly improved. This illustrates several important points:

- Response curves recovered from one combination of scene and Ladybug2 CCD can be readily applied to other scenes or Ladybug2 CCDs. This was confirmed by applying curves recovered from indoor scenes to images captured outdoors and vice versa.
- How good a curve "looks" is a good indicator of how accurate it is.
- The best performing curves look very linear over most of their range.

The last point led us to experiment with a linear response assumption, meaning that the digital pixel values are assumed to already be linearly related to irradiance. Extensive experiments have confirmed that the behaviour of colour-processed images is in fact linear or at least close enough to linear that it is better to assume a linear response than to use the response curves recovered from the images, which can be unpredictable. Therefore, our naive HDR creation simply consists of demosaicking and directly normalizing the images. However, we have built our HDR processing utility so that it is possible to use a response curve in case future hardware changes or additions to the image processing pipeline result in a non-linear response.

As a last note on calibration, since there is typically some unknown gain of the camera that converts actual electrons or photons to digital numbers [102], the calibration process

outlined so far is relative, not absolute. Absolute calibration can be achieved if a known illuminant is photographed, in which case the HDR values can be mapped back to an absolute physical quantity. We do not at present perform absolute calibration.



(a) (b)

Figure 3.2: An improved calibration result. a Robertson curve is now much more monotonic. b Tonemapped result. Note the reduction in banding.

3.5 Other Considerations

In addition to the naive HDR process detailed so far, one question is how to make the transition from grayscale to colour image assembly. The existing literature on this topic describes the situation for grayscale, and does not provide details on colour processing. Having completely independent weights for each channel can provide good results in many cases, but goes against the exposure-weighting concept. Since the relationship of a pixel's color channel values encodes its color, when one channel saturates before the others and therefore receives a lower weight than the others, the colour can change drastically, resulting in chromatic artifacts. Therefore, the weight should be a measure of how much an entire pixel contributes to the final composite and not how much each of its channels independently contribute. This effect can be seen in Figure 3.3a, where the sky is tinted yellow.

The chromatic artifacts are largely fixed by taking the average of the RGB weights for a given pixel as shown in Figure 3.3b. This scenario also reveals a second problem: when a pixel is fully saturated in all shots, the resulting output is grey since the same

value is divided by the exposure time of each image before averaging together. This effect is seen in the upper left corner of the image which looks directly at the sun.

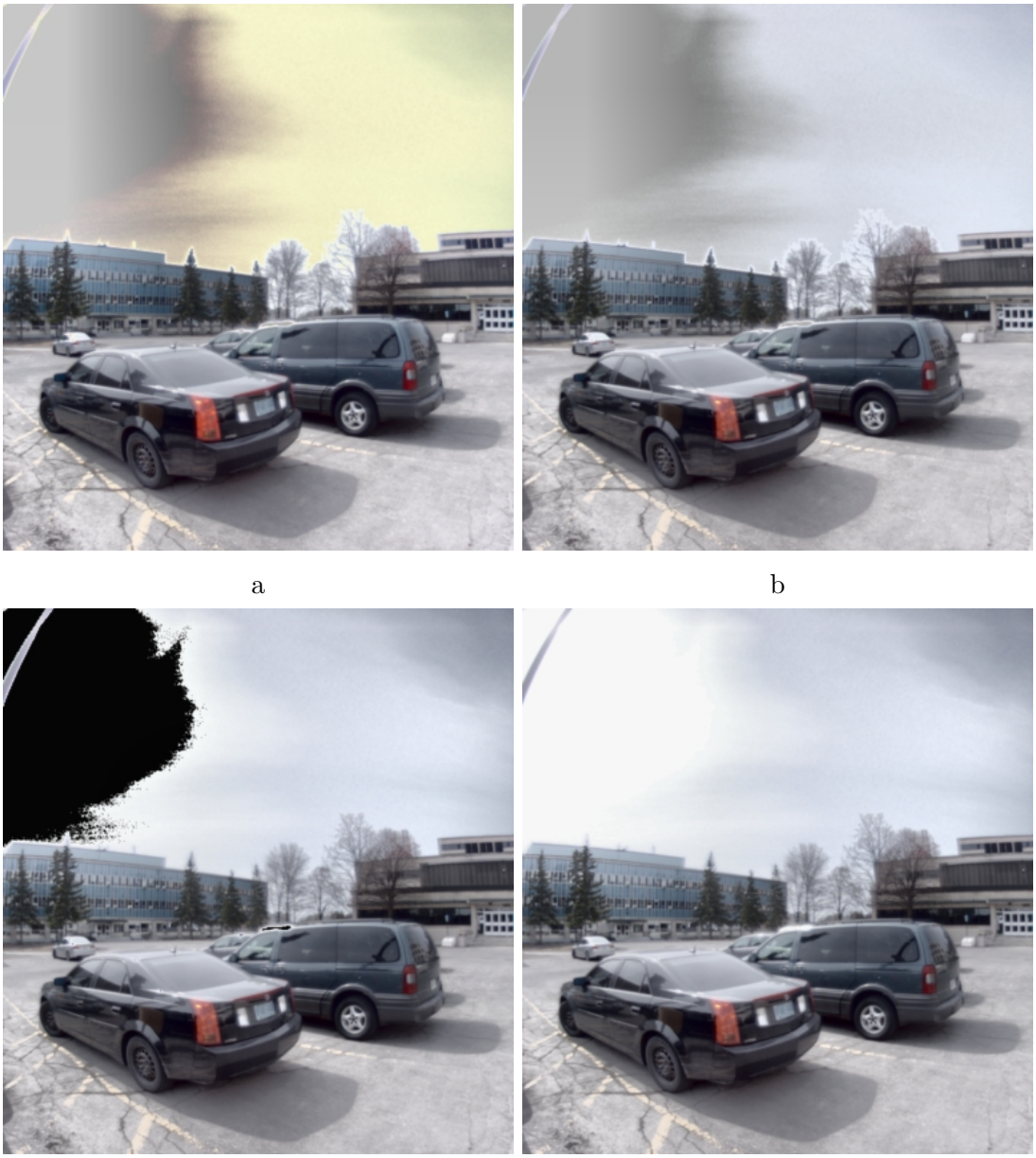
Since a pixel that is always saturated essentially tells us nothing, we can reduce the weighting function to zero for the extremes of the sensor's range; *pfscalibration* uses a conservative cutoff of 10, meaning that the 10 lowest and highest possible quanta of the image are given a weight of zero. In our experiments, we discovered that cutoffs as low as 1-2 still produced very good naive HDR results, though some subtle gray fringes could sometimes be seen around high-contrast edges, so we choose a midway cutoff of 5, where any pixels always over or under saturated result in NaNs (Not a Number - short hand for division by zero); the result is shown in Figure 3.3c. Finally, we note that the input LDR images allow us to check, for a given NaN pixel, whether it was over or under saturated througout the capture process, in which case a reasonable solution is to assign it the highest or lowest HDR pixel value, respectively, found in the non-NaN portions of the final result. The result of applying this final check is shown in Figure 3.3d, and results with these checks applied will also be referred to as naive HDR results since they still do not factor in motion.

The final question is which of the weighting functions discussed in Section 2.2.2 to actually use. We experimented with the hat [2], wide hat [30], and pseudo-Gaussian [7] weighting functions, and found performance similar for all. We use the pseudo-Gaussian weighting function of Robertson et al., but without the multiplication by exposure time.

3.6 Assembly of Panoramas

For storage purposes, the six views of the Ladybug2 must be reprojected into a panoramic image format such as a cubic, cylindrical, or spherical panorama. For this purpose, and for compatibility with the existing LDR pipeline upon which we are building, we use the approach and code of Bradley et al. [23], which allows remapping to a cubic panorama consisting of 6 cube faces corresponding to the four sides plus top and bottom of a cube.

Briefly, this remapping is accomplished by virtually orienting 6 cube faces centered around the Ladybug2 camera. Iterating through each pixel of each cube face, we take the coordinate of the pixel and use the Ladybug SDK functions *ladybugXYZtoRC* and *ladybugUnrectifyPixel* to map from real-world XYZ coordinates to rectified image coordinates and from there to unrectified (CCD) image coordinates respectively. When the resulting coordinate does not fall precisely on a single pixel of a CCD, the value is interpolated from nearby pixels using bilinear interpolation. When it falls in the overlap



a b

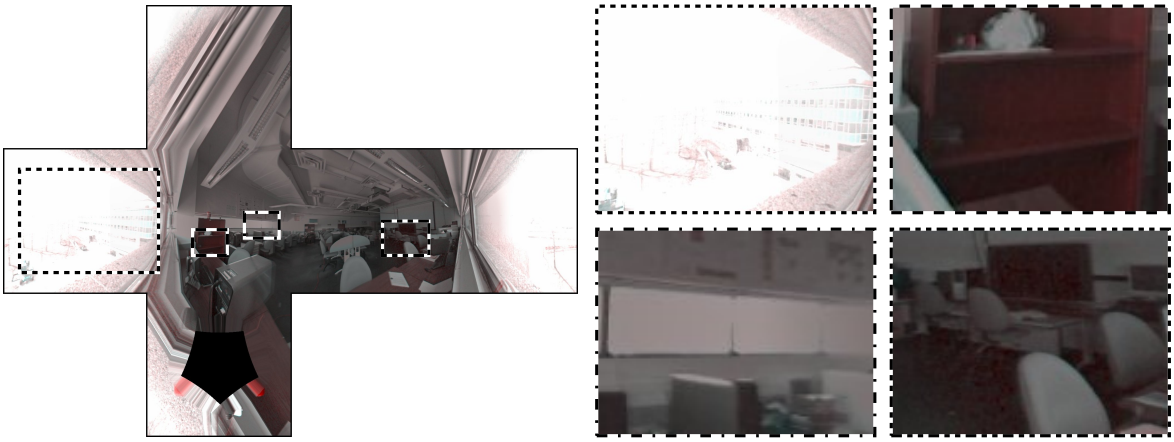
c d

Figure 3.3: Special considerations in weighting. a Weighting function applied independently for each channel. b Weights averaged across channels. c Same as previous but weights reduced to zero for extreme values. d Same as previous but poorly exposed pixels clamped to min or max of well exposed pixels.

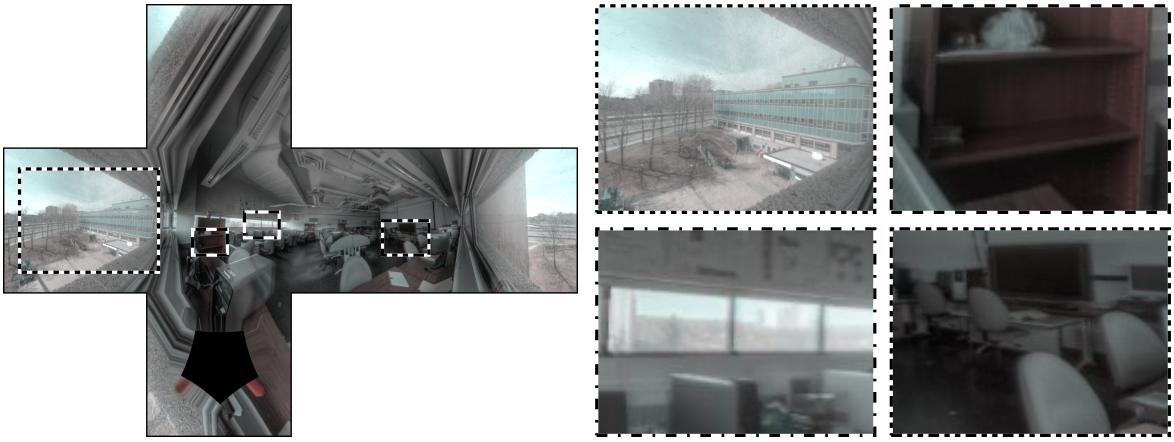
between two CCDs, the values of the CCDs are blended using alpha masks retrieved from the camera. When it lands outside of all CCDs, the value defaults to black. See [23, 103] for more details.

For a naive HDR pipeline, we create independent HDR images for each CCD view, then apply the same remapping process to create an HDR cubic panorama. While our capture code locks the gain for a given CCD and varies exposure using exposure time only, separate CCDs may have separate gains, so CCDs for a given panorama are normalized to a common domain by dividing by Δt *and* gain, G.

Figure 3.4 shows a comparison of an LDR pipeline approach and the proposed HDR approach. The LDR result of Figure 3.4a, which might be considered semi-HDR is mentioned by Reinhard et al. [30] and detailed by Bradley et al. [23]. We note that even with all CCDs in independent auto-exposure mode, it is easy to exceed the dynamic range of individual CCDs, with the result that bright regions are saturated, and dark regions are noisy. By instead applying an HDR approach on each CCD individually, each CCD can capture considerably more of the range of the portion of the scene it sees. The result after remapping is a panorama with greatly expanded dynamic range as shown in Figure 3.4b. Note that these two results are displayed with different tonemapping operators. The LDR pipeline tested integrates the bilateral filter-based tonemapping of [89], while we use the Details Enhancer mode of Photomatix Pro's tonemapper for good visibility of dark regions.



(a)



(b)

Figure 3.4: Stitched HDR panorama result. a Result of LDR pipeline, obtained by normalizing independently auto-exposed CCD images then tonemapping with bilateral filter before remapping. b Result of proposed naive HDR panorama approach, obtained by creating naive HDR images for each CCD before remapping and tonemapping.

Chapter 4

Background Estimation

In real-world image capture scenarios, particularly for panoramic capture aimed at telepresence applications, it is desirable to remove moving objects in the scene. One reason is the privacy concerns related to large scale capture of public environments, evidenced by publications from the legal domain [104, 105]. Various methods have been proposed to mitigate privacy concerns [106, 107], while Google now uses automated processing to blur faces and license plates, and offers a tool that members of the public can use to report areas of images they would like further blurred[1]. In addition, if the captured environment is to be later augmented with virtual objects, e.g. a 3D rendering of a proposed new building or rendered characters for a video game application, it is helpful to start with a background model of the scene.

We therefore approach HDR deghosting as a background estimation problem. For illustrative purposes, we outline in this chapter a LDR version of our background modeling. In Chapter 5, we describe the HDR version, which introduces new challenges but also opens avenues for further analysis since we have changing exposure settings which allows us to make assumptions about how pixel values should change between images.

Capturing a series of images of the same scene over a period of time enables background estimation techniques like those outlined in Section 2.3.3. If we consider a set of identically-exposed images of a scene with only background objects, and no system noise so that all images are identical, then the background modeling problem is solved by simply taking any one of the input images as the model. If we allow for system noise such as sensor read noise and quantization error, then averaging the images together as shown in Section 2.2.2 will provide a slightly improved background model, particularly

[1] http://maps.google.com/intl/en_us/help/maps/streetview/privacy.html

if the images are captured with high gain and/or contain dark regions.

Now, if we take a similar image set of the same scene but captured while objects in the scene moved, as in Figure 4.1, then averaging the images will result in ghosting. However, if we can determine that an area in a given image represents a moving object, then it can be removed from the composite by introducing weights in the average and reducing these to near zero in that area. In this chapter we outline a method for accomplishing this task in the case of non-exposure-bracketed images.

Our method is designed to integrate into a large-scale image capture project, and could at times be used to process hundreds or thousands of images, so computation time is a limiting factor. Therefore we choose a change-detection approach over optical flow or graph cut methods. We start with a simple form of inter-frame change detection to detect motion throughout the image sequence. These regions are down-weighted on a per-frame basis so that they contribute very little to the final composite, allowing us to remove ghosts while still maintaining the noise removal advantages of averaging multiple exposures. We also demonstrate the use of state of the art superpixel oversegmentation techniques for refining detected areas, an improvement on the simple morphological operators popular in previous approaches. Finally, we introduce a novel approach for detecting and freezing fluid motion areas, those that have seen motion throughout the capture process since they will not be fixed by the per-frame down-weighting alone. Several parameters are introduced in this chapter and the next. The effect of varying them will be discussed in Section 6.3.

4.1 Change Detection

Section 2.3.2 provided a brief introduction to change detection which we expand on here. Considering the images of Figure 4.2, in a machine vision application, we would like to know what areas have changed between the two images. If we consider two grayscale images, $\mathbf{Z}_m$ and $\mathbf{Z}_n$, a mask[2] of areas changed between the images can be created by taking their absolute difference and keeping all values therein that are greater than some threshold value:

$$\mathbf{M}_C^{Basic} = \{(i,j) \mid |z_{i,j,m} - z_{i,j,n}| > t_C\}, \tag{4.1}$$

[2]A mask, in the image processing context, is a matrix the same size as an image of interest, with ones markings areas of interest based on some criteria, and zeros elsewhere. In practice, they are implemented as such, but here we denote them as sets of pixel indices corresponding to the areas of interest.

Figure 4.1: A sequence of LDR images of the same scene displaying motion corruption.

where (i, j) index pixels, m and $n = m + 1$ index N input images, and t_C is some cutoff value. One question is how to choose t_C. Otsu's method [44] calculates the threshold automatically by assuming that the image contains two classes defined by two distributions in the image histogram, then minimizing the weighted sum of the intra-class variances [108]. This works well when the assumption of two pixel classes holds, as in controlled machine vision scenarios such as product inspection on an assemly line; one image can be captured while the assembly line is empty and the second can be captured at run time when it is known that a new object is somewhere on the line. In scenarios where it is not guaranteed that anything has changed between the images, we found it best to use a fixed percentage threshold, as Otsu's method can sometimes force a two-distribution model when it is not warranted. So, Expression 4.1 becomes:

$$\mathbf{M}_C^G = \left\{ (i,j) \left| \frac{|z_{i,j,m} - z_{i,j,n}|}{z_{i,j,m}} > t_C \right. \right\}, \tag{4.2}$$

Colour images provide more information for change detection algorithms to work with. While research into optimal colour spaces for change detection has been done [109,110], it is important at this point to consider the differences between our goal and those of most change detection systems. Typically, change detection aims to find approximate outlines of moving objects at high speeds for the purpose of triggering some other process, such as an alarm in a surveillance application. As such, it is not a problem if some of the object is missed. Research into improving change detection has often focused on reducing over-detections due to shadows [110]. Since our approach is part of a graphics pipeline, we would rather err on the side of over-detection rather than under-detection since if we over-detect, the deterioration in image quality should be minimal, but underdetecting can lead to keeping parts of moving objects in the background model. Regarding shadows specifically, we would like to count them as a change so that they are removed from the background as well. In our experiments to date, we apply the above-discussed change detection separately to each channel, k, of two RGB images, then consider a change in any of the channels as signalling a change:

$$\mathbf{M}_C = \left\{ (i,j) \left| \bigvee_{k=0}^{2} \left(\frac{\left| z_{i,j,k,m}^{F_{avg}} - z_{i,j,k,n}^{F_{avg}} \right|}{z_{i,j,k,m}^{F_{avg}}} > t_C \right) \right. \right\}, \tag{4.3}$$

where $\mathbf{z}^{F_{avg}}$ is $\mathbf{z}$ after a 3×3 average filtering, which was found to improve the basic change detection, particularly in dark regions which tend to be noisy.

Figure 4.3 shows an example of applying change detection to the images of Figure 4.2. In the first example, Otsu's method is used to threshold the absolute difference of grayscale versions of the images. In the second example, Otsu's method is applied independently on each channel. Notice that it provides an improvement on the masks. In the third example, we apply a percentage threshold. With the percentage threshold, there is a lot more of what would be considered simply noise in many other background modeling applications due to moving foliage. We will ignore this for the current discussion but, as we will show, we do in fact want to detect moving foliage and doing so will be integral to our fluid motion fix introduced at the end of this chapter.

At this point, we have regions with some holes in them. While these regions are likely sufficient already for many surveillance type applications, they will not work well for our application since a small hole in the region means part of it will be kept. While the masks

a b

Figure 4.2: Object motion in two consecutive LDR images.

a b c

Figure 4.3: Basic change detection on the images of Figure 4.2. a Change detection applied to grayscale versions of the images using Otsu's method to threshold. b Change detection applied to each channel using Otsu's method. c Change detection applied to each channel using a percentage change threshold $t_C = 0.40$.

can be cleaned up using morphological operators, in our HDR deghosting scenarios, we often found this was insufficient, so we instead apply a segmentation algorithm to find salient boundaries in the image, then use the change mask to select regions which become the refined change mask.

4.2 Segmentation

In typical change detection results, it is difficult to produce an accurate and complete mask directly from the change detection step. More often, parts of the moving object

are missed, and parts of the background are mistakenly included in the detection of the object. In most existing approaches, various filtering and morphological operators are used to clean up the mask outlining the moving object. A popular approach is to use dilation followed by erosion to bring together disconnected blobs, then remove small disconnected blobs. The problem with morphological operators is that they are applied directly to the mask without factoring in information from the image itself, and as such do not respect boundaries in the image. An alternative to morphological operators is to use the change detection mask as a controller to select regions of a segmentation of the images, which do respect boundaries. We are not currently aware of existing methods specifically for HDR deghosting (which we address in Chapter 6) that use segmentation algorithms instead of simple morphological operators.

The first approach explored is marker-controlled Watershed segmentation, outlined in Section 2.4.1. Figures 4.4a and b show a grayscale version of the second image from Figure 4.2, along with the change mask found previously. What we would like is to separately find a meaningful segmentation of the image into regions, and use the change mask to select regions as foreground. This is proposed as an alternative to morphological operations on the change mask, which do not consider details of the original image.

Watershed segmentation is often performed on the gradient of the image, which is shown in Figure 4.4c. However, since there are many local minima, the result of a Watershed segmentation directly on the gradient is severe oversegmentation. This can be mitigated by applying an H-minima transform to the gradient to remove all minima with a depth less than some chosen threshold [63]; d shows the result with minima shallower than 55 removed, where this value was chosen by trial and error to provide a good segmentation of the person. It is immediately obvious that the segmentation is very uneven across the image, and it is unclear how we would choose the threshold automatically to properly segment objects of interest.

Another approach is to use a minima imposition algorithm to ensure that regional minima only exist in desired areas, a process known as marker-controlled Watershed segmentation [68]. In this case, we use morphological operators to clean up the change mask slightly, then erode it and dilate it to create foreground and background markers respectively (e). These are then used to impose minima on the gradient of the image, as shown in f. The result of running the Watershed algorithm on this new gradient image is shown in g, where we have used the mask of b to select as foreground any regions in which 25% of the region's area was detected as moving.

After testing this method on multiple test sets (most more challenging than the

example shown), we determined it was insufficient for our purposes. Even in this example, we can see in Figures 4.4h and i that this technique misses parts of the moving object.

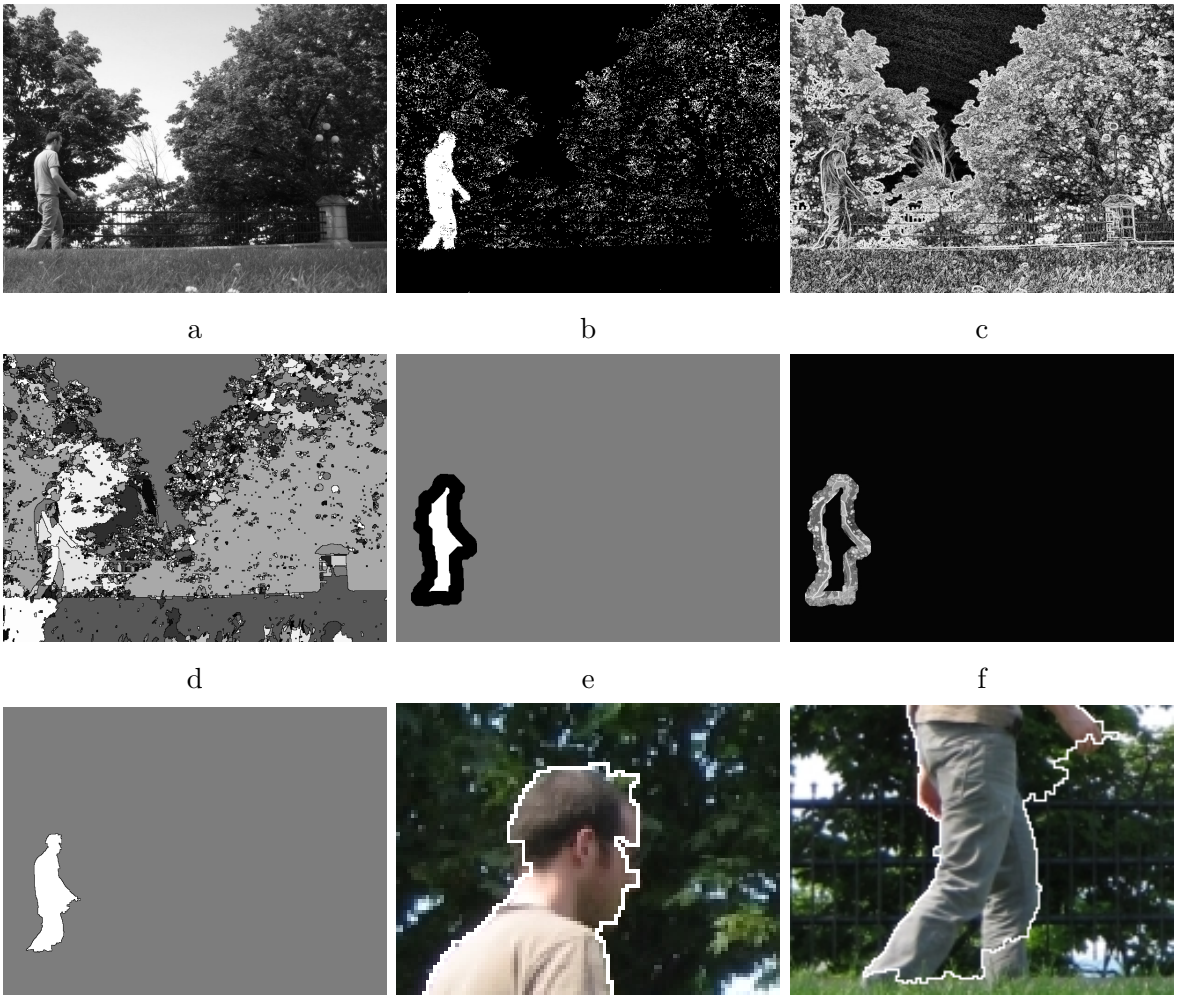
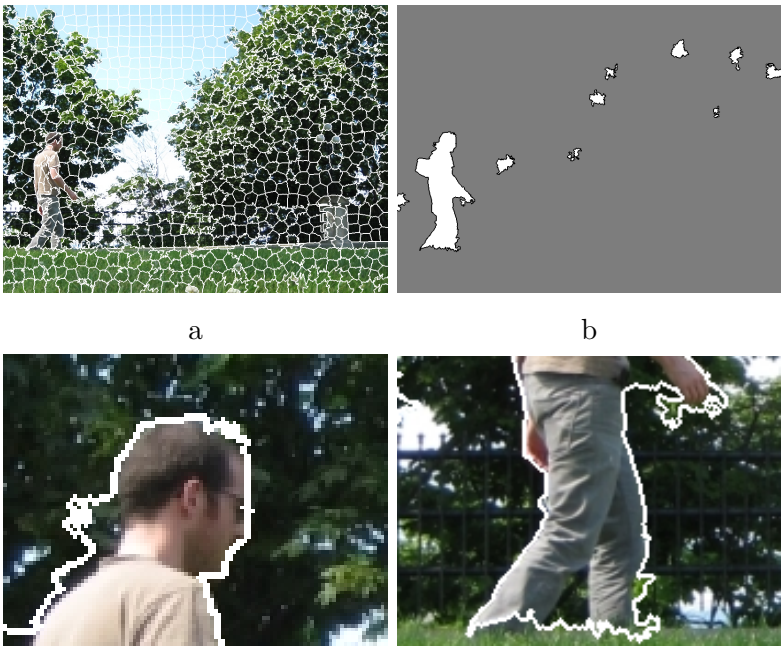


Figure 4.4: Mask refinement by marker-based Watershed segmentation. a Grayscale input image. b Moving object mask to be refined. c Gradient magnitude of input image. d Result of applying the H-Minima transform to gradient before applying Watershed [63]. e Foreground (white) and background (gray) masks after morphological operators. f Gradient after minima imposition [69]. g Foreground and background regions refined. h Close-up of head. i Close-up of legs.

Recent superpixel methods [75–77], reviewed in Section 2.4.2, offer intentional and controlled over-segmentation of an image, with a high level of respect for local image boundaries. Figure 4.5a shows a segmentation of our test image using the SLIC Su-



a b c d

Figure 4.5: Mask refinement by superpixel segmentation. a Superpixel outlines. b Foreground and background regions refined. c Close-up of head. d Close-up of legs.

perpixel algorithm [77]. The segmentation shown has about 350 regions. Like the Watershed example, we assume that any regions seeing 25% or more moving pixels are moving objects. More formally, if the superpixel algorithm returns a superset $\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_{N_{SP}}\}$, where N_{SP} is the number of superpixels and each subset $\mathbf{S}_p$ is the set of pixel coordinates (i, j) masking superpixel p, then a refined change mask is given by:

$$\mathbf{M_{CR}} = \left\{ \mathbf{S}_p \,\middle|\, \frac{|\mathbf{S}_p \cap \mathbf{M}_C|}{|\mathbf{S}_p|} > t_{SParea} \right\}, \tag{4.4}$$

Figures 4.5b-d show the results, and we can see that while more of the background has been detected as moving, none of the actual moving object has been missed, which is the most important consideration for our system since we are trying to figure out what regions to reject from an averaging of multiple images when creating the background model. We also see that though the foliage was quite still, some parts moved enough to select additional superpixels aside from the person. This is not problematic and in fact will be important for our fluid motion fix introduced later in this chapter.

4.3 Down-weighting of Moving Regions

If we return to the concept of the background model as a weighted average of the input images, and consider the moving regions detected after superpixel refinement in the previous section, these areas can be removed from the final composite by reducing their weight for that particular image to a very small amount. One catch is that we do not know which of the two images of Figure 4.2 actually contains the moving object, just that the mask of Figure 4.5b tells us what regions changed between those two images. Similarily, if an object *moved* between shots rather than appearing, it would create two motion regions: one where it departed and one where it arrived. Without higher level analysis such as optical flow, feature tracking, or scene characteristic analysis, it is impossible to determine which is which. However, if we assume we have enough input images that the region in question is seen multiple times, we can down-weight a given region in both images currently under inspection. The only downside is slightly less noise reduction in the LDR version, or a slight reduction in dynamic range in the HDR version that we will describe later.

An example of weight matrix modification is shown in Figure 4.6, where we see the results of comparing frames 1 and 2 of Figure 4.1. As before, the difference of the two images is calculated in colour, and is only shown in grayscale here for illustrative purposes. It is thresholded separately for each channel and for any given pixel, a change in any channel is counted as an overall change. The mask is then refined with Superpixel segmentation. Finally, the weight matrix for both images currently being inspected is reduced to near-zero in the regions outlined in the refined change mask . We also apply a slight fade in the weight matrix around the change regions to avoid sharp transitions in the final composite image. Figure 4.6d shows the weight matrix of the first image after this comparison. Of course, as we proceed to compare frames 2 and 3, and motion is detected in different areas, the weight matrix of image 2 is modified a second time, resulting in the weight matrix of Figure 4.6e

Repeating this process of inter-frame comparisons and down-weighting of moving areas, which we will refer to by itself as pair-wise down-weighting, we obtain the result shown in Figure 4.7; comparing this to the basic averaging result of Figure 4.7a, we can see that the ghosts have been removed. Pseudocode for the steps described so far is provided in Algorithm 4.1. In the following section we outline a secondary process to handle fluid ghosts, the regions that see motion repeatedly throughout the image sequence.

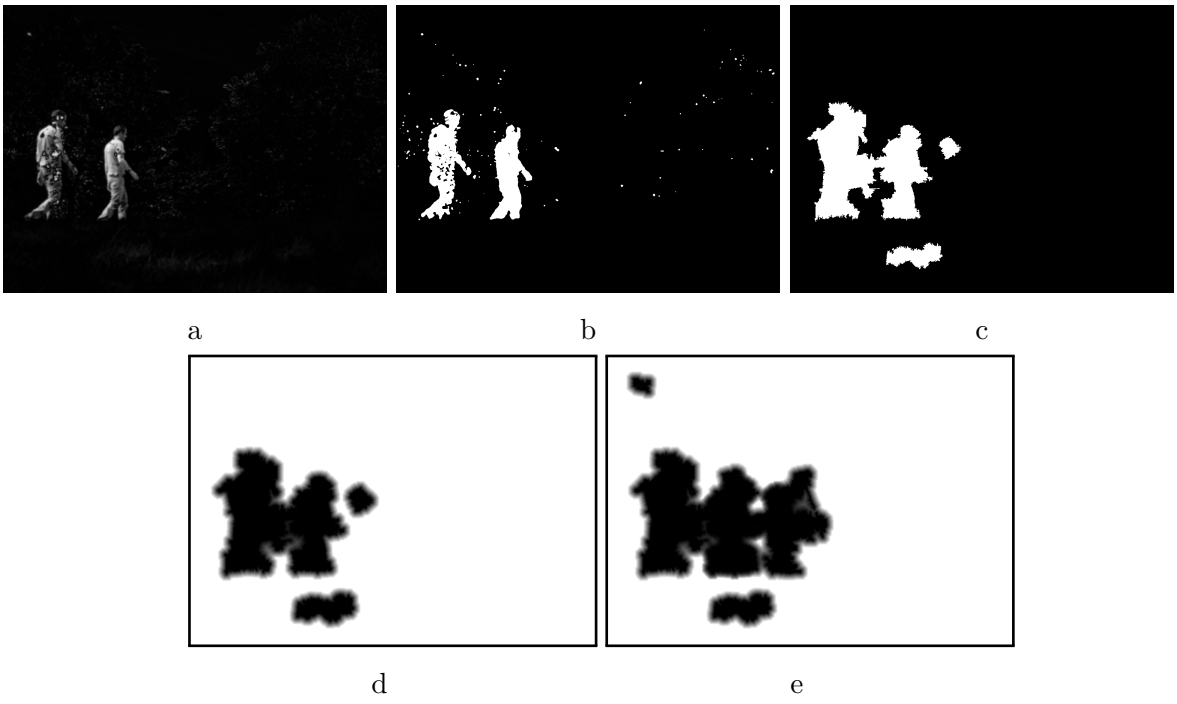


Figure 4.6: Weight modification for background estimation. a Grayscale absolute difference of two frames. b Difference map. c Refined difference map. d Image 1 weights modifed. e Image 2 weights modifed after next comparison.

4.4 Areas Experiencing Fluid Motion

One scenario in which the process outlined so far fails is when a portion of the scene sees motion repeatedly or continually throughout the capture process. This can result with visually fluid types of moving objects such as foliage and flags blowing in the wind, or waves on water. This can also result when discrete ghosts like people stay in one place and deform, e.g. if a person stays in one place but varies their pose. We will use the term "fluid" as a catch-all term to describe any type of ghosting resulting in repeated change detection in one location, whether due to objects that are actually fluid looking (waves, foliage, flags), or due to discrete objects deforming or otherwise triggering change detection repeatedly in one local region of the image.

For truly fluid objects like waves, trees, and foliage, it is unclear what exactly counts as the background, and arguably any single shot is a valid model of the background. This re-illustrates the difference between our application and more common machine vision or computer vision background modeling applications where the model would incorporate

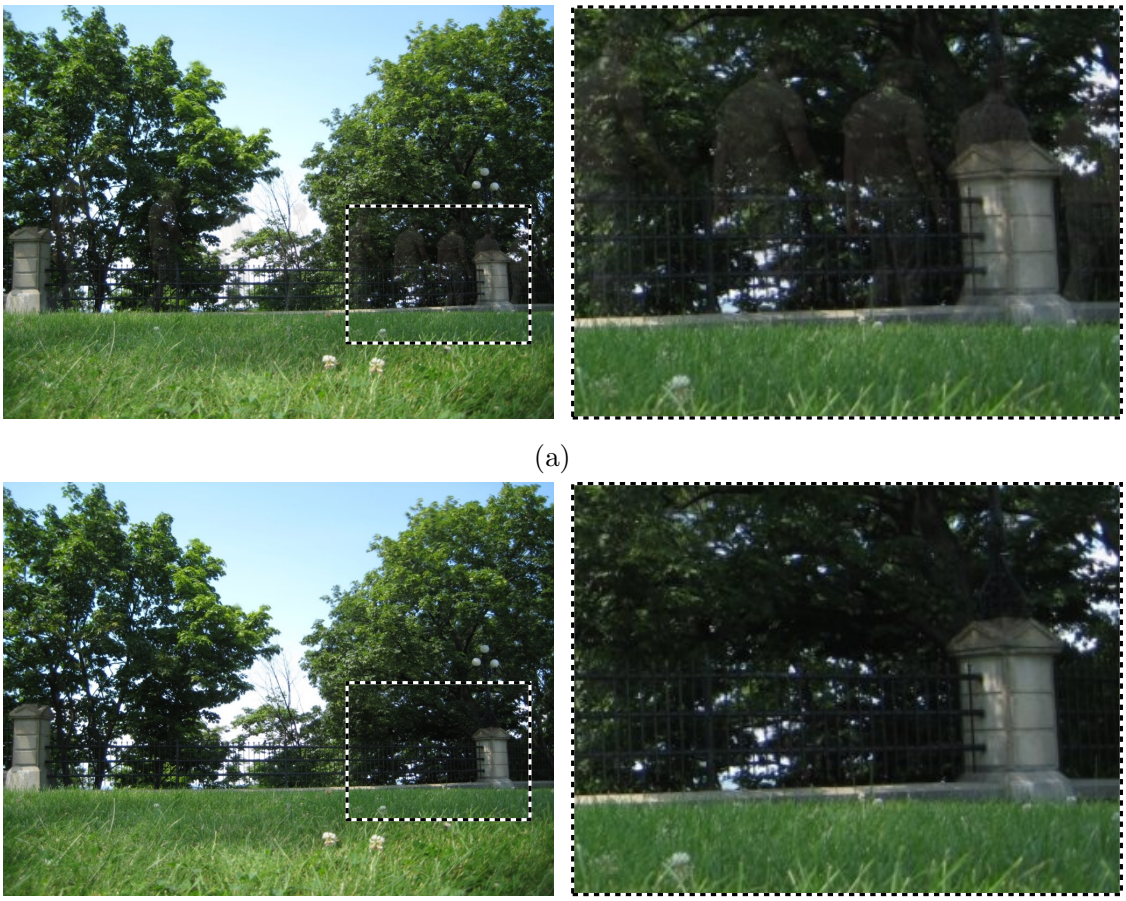


Figure 4.7: LDR pair-wise down-weighting result. a Naive result (basic average), along with a magnified portion showing ghosting. b Deghosted result, with magnified portion showing reduced ghosting.

the fact that some pixels oscillate between multiple states which all count as background. On the other hand, when a stationary discrete object like a person varies its pose, we can reasonably claim that it is not part of the background; however, if the background is rarely or never seen, then often the best we can do is choose a single input image to "freeze" the ghost in one pose. Ultimately, in both cases we need to choose a single input image as representative of the effective background, with the main goal being visual plausibility, which is why we will refer to both scenarios as fluid motion.

Figure 4.8 shows an example ghosting scenario with several areas experiencing fluid motion. The result of naively combining these images is shown in Figure 4.9, and the result of the pair-wise down-weighting process described in the previous section is shown

Algorithm 4.1 Background Estimation for LDR Images

Change Detection

Load N input images of size $W \times H$

Create N weight matrices of size $W \times H$, intiliaze to 1.0

$\mathbf{M}_A = \{\}$ ▷ Mask to track what areas see change throughout frames

for Each pair of consecutive images, $\mathbf{Z}_1$ and $\mathbf{Z}_2$, **do**

    Detect changed areas, $\mathbf{M}_C$ (Expression 4.3)

    Segmentation

    $\mathbf{M_{CR}} = \{\}$ ▷ Refined change mask for current image pair

    for Each of $\mathbf{Z}_1$, $\mathbf{Z}_2$ **do**

        Run Superpixel segmentation on $\mathbf{Z}$ to get $\mathbf{S}$

        $\mathbf{M_{CR}} = \mathbf{M_{CR}} \cup \left\{ \mathbf{S}_p \,\middle|\, \frac{|\mathbf{S}_p \cap \mathbf{M}_C|}{|\mathbf{S}_p|} > t_{SParea} \right\}$

    end for

    Down-weighting

    for Each of $\mathbf{Z}_1$, $\mathbf{Z}_2$ **do**

        Reduce weight matrix for $\mathbf{Z}$ where marked by $\mathbf{M}_{CR}$

    end for

    Feather edges of newly down-weighted areas

    $\mathbf{M}_A = \mathbf{M}_A \cup \mathbf{M}_{CR}$

end for

Compose background image $\mathbf{Z}_{BG}$ using modified weights

Fluid Motion Fix (see Algorithm 4.2)

in Figure 4.10. Pair-wise down-weighting has removed many of the ghosts, but several areas have ghosts remaining because they saw fluid motion.

To improve these problematic areas, we wish to first find these areas, then decide on the best input image to replace them. To find the fluid motion areas, we can make use of information collected during pair-wise down-weighting. Since we calculate change masks frame to frame, these can be accumulated into an overall change mask that highlights all regions that saw motion at any point during the capture process, as shown in Figure 4.11a.

Next, we sum the inter-frame change masks across all input images to find out how many times a given pixel saw motion during the capture, as shown in Figure 4.11b. Then, we apply a threshold, t_{MC}, to the number of times motion is seen as a fraction of the number of input images. In our testing so far, we have used a threshold of 25%,

Figure 4.8: A series of LDR images of a scene displaying fluid motion in some regions.

so that any pixel that sees motion in 25% or more of the input images counts as fluid. Figure 4.11c shows the pixels counted as fluid overlaid with the accumulated change mask, which has been eroded to remove small connections between regions; areas where the two coincide are shown in green. The resulting separate regions that have always seen motion and need to be fixed are shown in Figure 4.11d, where regions are distinguished using connected component labeling [62, Sec.9.4].

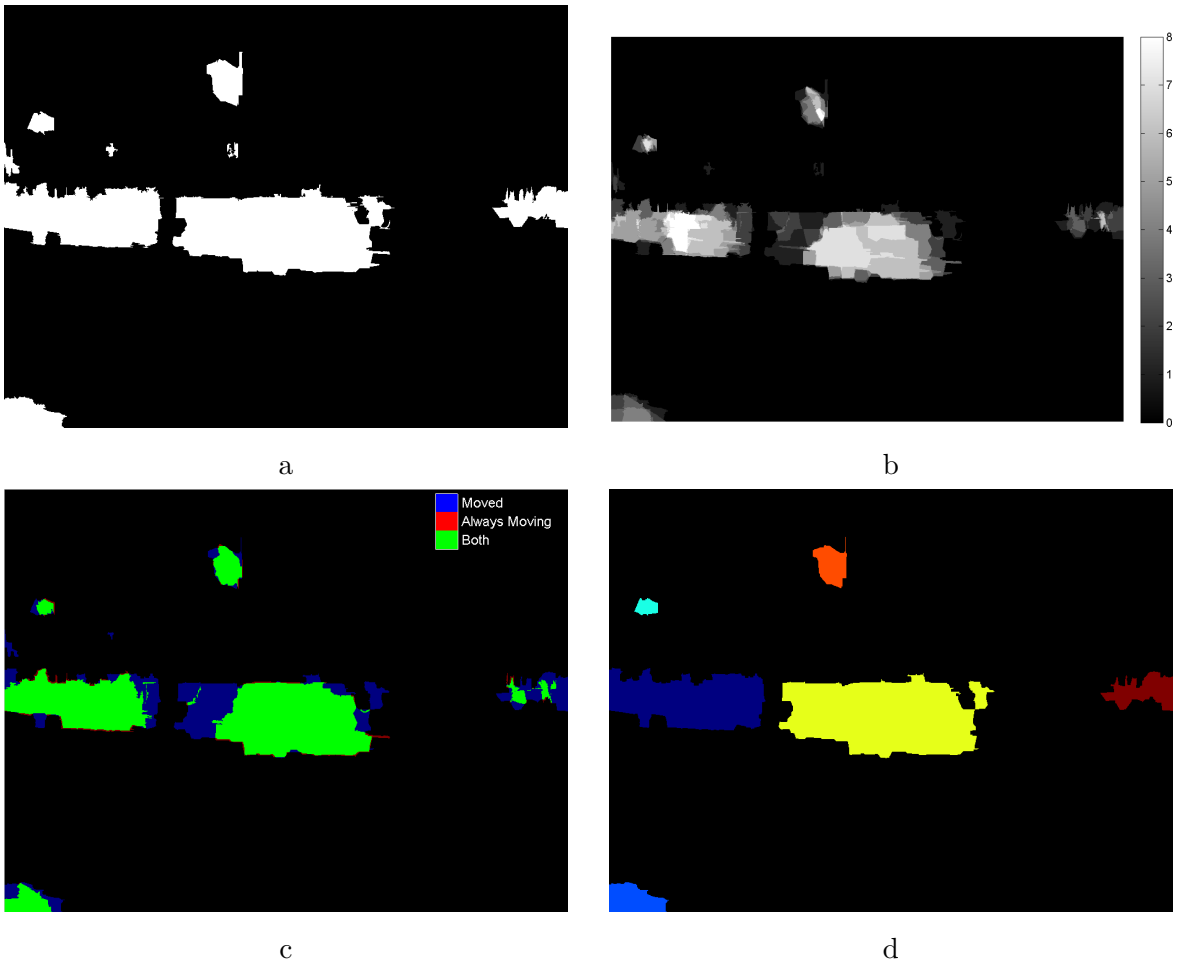
Next, we proceed through each of the regions and find the best candidate to replace the region. Since the best candidate would be that which looks the most like the background, we use the naive result as an approximate background. For the region currently under inspection, the mean squared error (MSE) of each of the input images with respect to the naive result is calculated, and the input image giving the minimum MSE is chosen. The region is then replaced with information from that single input image, and a basic center-weighted average blending is applied around the edges to reduce sharp transitions, a consideration which will become more important in the HDR version. See Szeliski for details on this blending technique [111, Sec.6.2], and Section 6.1 for further discussion on blending and associated artifacts.

The result of applying our fix for areas experiencing fluid motion is shown in Figure

Figure 4.9: Naive result for areas experiencing fluid motion, with magnifications of corrupted areas.

Figure 4.10: Result of pair-wise down-weighting for a set with areas experiencing fluid motion, with magnifications of corrupted areas.

4.12. In the case of the flags, pair-wise down-weighting had improved them somewhat, but still left them blurry. They were detected as fluid and replaced with the best match for the background as approximated by the naive result, and are now much more crisp. The lower left close-up shows an area that experienced motion throughout the capture process from several different sources: cars, people, and bicyclists. Here, the best match found contained two people, but does not display any ghosting behaviour. The lower right close-up is an interesting scenario in which two people stayed in one small region but moved about and changed poses during capture, making pair-wise down-weighting fail. Again, a best replacement was chosen from the input images and we have now fixed



a b c d

Figure 4.11: Regions experiencing fluid motion. a All areas that saw motion at any point during capture. b The frequency with which motion was seen at each pixel. c Regions that saw motion during capture and those that always saw motion. d Result of thresholding regions that saw motion.

the people at one instant in time and reduced the ghosting. Close inspection reveals a tiny remaining ghosting artifact near the feet of the girl on the left, but overall this represents a big improvement over pair-wise down-weighting alone. Pseudocode for the fluid motion fix is provided in Algorithm 4.2.

Figure 4.12: Result with proposed fix for areas experiencing fluid motion, with magnifications of corrupted areas.

Algorithm 4.2 Fluid Motion Fix

Continued from Algorithm 4.1

Erode $\mathbf{M}_A$ and do connected component labeling to get $\mathbf{M}_{AE} = \{\mathbf{M}_{AE_1}, \ldots, \mathbf{M}_{AE_R}\}$

Sum all inter-frame change masks to get matrix $\mathbf{S}_{MovingCount}$

$\mathbf{M}_{FluidMotion} = \left\{(i,j) \left| \frac{\mathbf{S}_{MovingCount}(i,j)}{N} > t_{MC} \right.\right\}$

$\mathbf{M}_{FluidMotionRegions} = \left\{\mathbf{M}_{AE_r} \left| \frac{|\mathbf{M}_{AE_r} \cap \mathbf{M}_{FluidMotion}|}{|\mathbf{M}_{AE_r}|} > t_{MFA} \right.\right\}$

for all Regions $\mathbf{M}_{CurRegion}$ in $\mathbf{M}_{FluidMotionRegions}$ **do**

    Replace $\mathbf{M}_{CurRegion}$ in $\mathbf{Z}_{BG}$ with input image having lowest MSE wrt naive BG model

    Feather edges of replaced region

end for

4.5 Summary

The combination of the pair-wise down-weighting described in Sections 4.1-4.3 and the fix introduced in Section 4.4 for areas seeing fluid motion allow us to handle two distinct types of ghosting: discrete and fluid, each with their own challenges. In the case of discrete ghosts, they are removed from individual frames that they affect, while the rest of the frames remain intact; this is advantageous in the HDR version introduced in Chapter 5. In the case of fluid ghosts, a best single input image is chosen to replace them and thereby eliminate ghosting; although this means that in some cases the background itself is not modeled, the improvement in visual plausibility over pair-wise down-weighting

alone is pronounced. With the addition of this new fix, in most scenarios, we produce results comparable to state of the art graph-cuts methods but with significant savings in computation time. Pseudocode for the approach introduced in this chapter is presented in Algorithms 4.1 and 4.2. Several parameters have been introduced in this chapter, and a discussion of their values for the HDR case is given in Section 6.3.

Chapter 5

Ghost Removal for HDR Imaging

In Chapter 4, we introduced a method to perform background estimation based on multiple captures of the same scene for an image-based rendering application. We also introduced a special check for "fluid motion" areas.

We now extend this approach to exposure-bracketed images, like those of Figure 5.1, for the purpose of ghost removal in HDR images. The result of naively combining exposure-bracketed images containing scene motion is shown in Figure 5.2. Working with exposure-bracketed images means we cannot directly apply basic change detection, which is the critical first step of our background estimation, so we must normalize the images first. We must also check several special cases before applying change detection since saturation affects change detection. However, knowing the exposure values of the images also allows us to make assumptions about pixel behaviour between images, so we can introduce new constraints that can capture additional movement.

5.1 Change Detection

In the previous chapter, change detection was applied directly on pairs of consecutive images. With exposure-bracketed images, we need to first normalize LDR pixel values, $\mathbf{Z}$, to bring them into a common illumination space so that a given pixel that has experienced the same irradiance at the sensor in two given images and was neither under- nor over-saturated in both images has the same value.

If the overall camera response function, $g(\cdot)$, is non-linear, then we start by converting digital pixel values to exposure values, $\mathbf{X} = g(\mathbf{Z}) = \mathbf{E}\Delta t$. We then normalize, i.e. convert to Irradiance, E, by dividing by exposure time Δt. More detail on this process

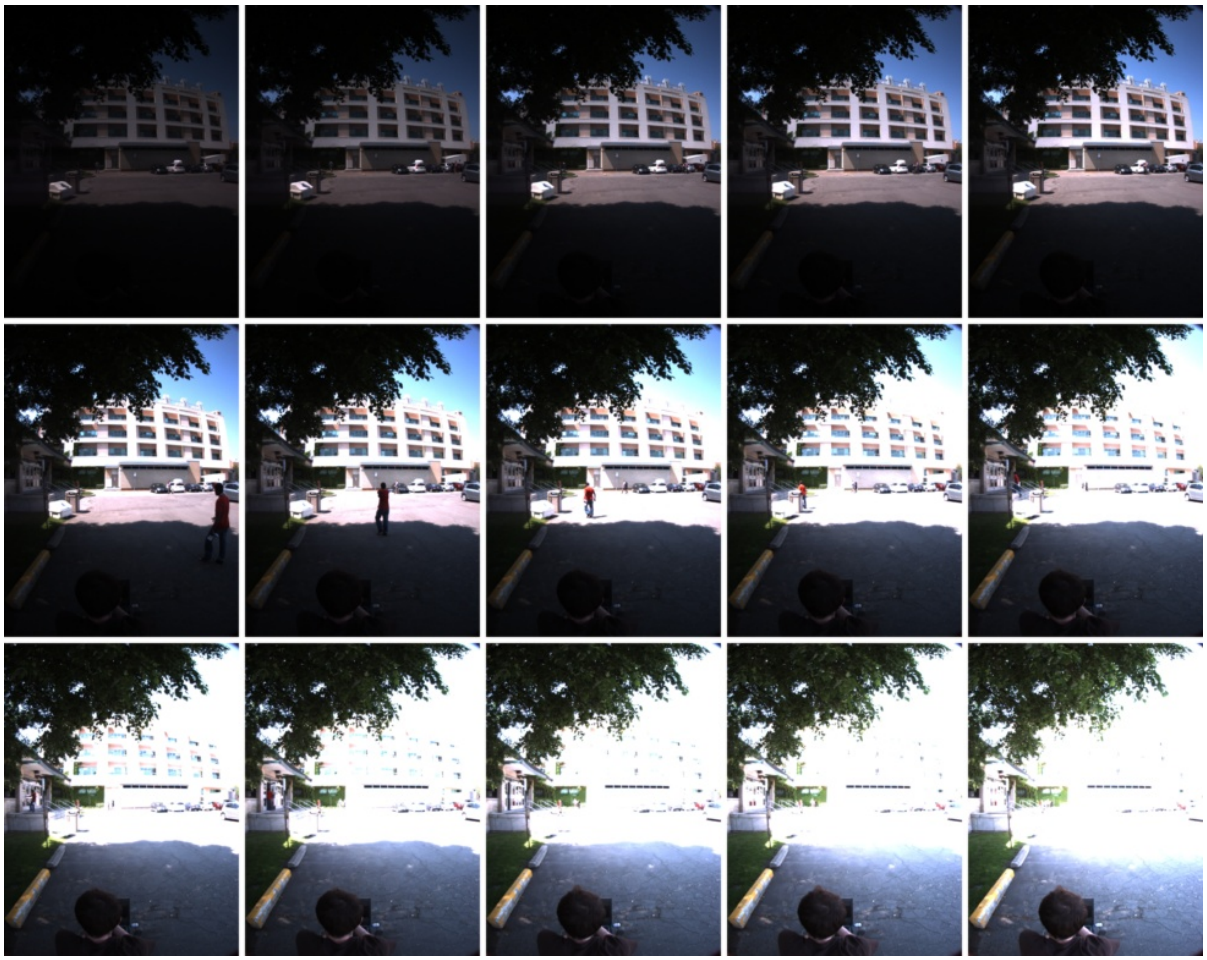


Figure 5.1: 15 of a 16-image sequence of exposure-bracketed images of the same scene displaying motion corruption. The scene is corrupted by multiple people walking through (discrete ghosts), as well as a large section of foliage blowing in the wind (fluid ghosting).

is provided in Section 2.2.1. As shown in Section 3.4.2, the Ladybug2's response after colour processing is linear, so the first step can be skipped and we can directly normalize the images. If gain is varied as well, then we replace Δt with ΔtG.

In the previous chapter, we proposed starting with weight matrices all initialized to 1. In the HDR scenario, the weight matrix for each image is initialized by applying the HDR weighting function to the LDR image as discussed in Section 2.2.2. We use Robertson's Gaussian weighting function [7].

Considering a pixel in two consecutive normalized images, $\mathbf{E}_1$ and $\mathbf{E}_2$, if it was well exposed in both associated LDR images, $\mathbf{Z}_1$ and $\mathbf{Z}_2$, it can be directly compared in $\mathbf{E}_1$ and $\mathbf{E}_2$ and considered to have seen a moving object in one of these images if its value

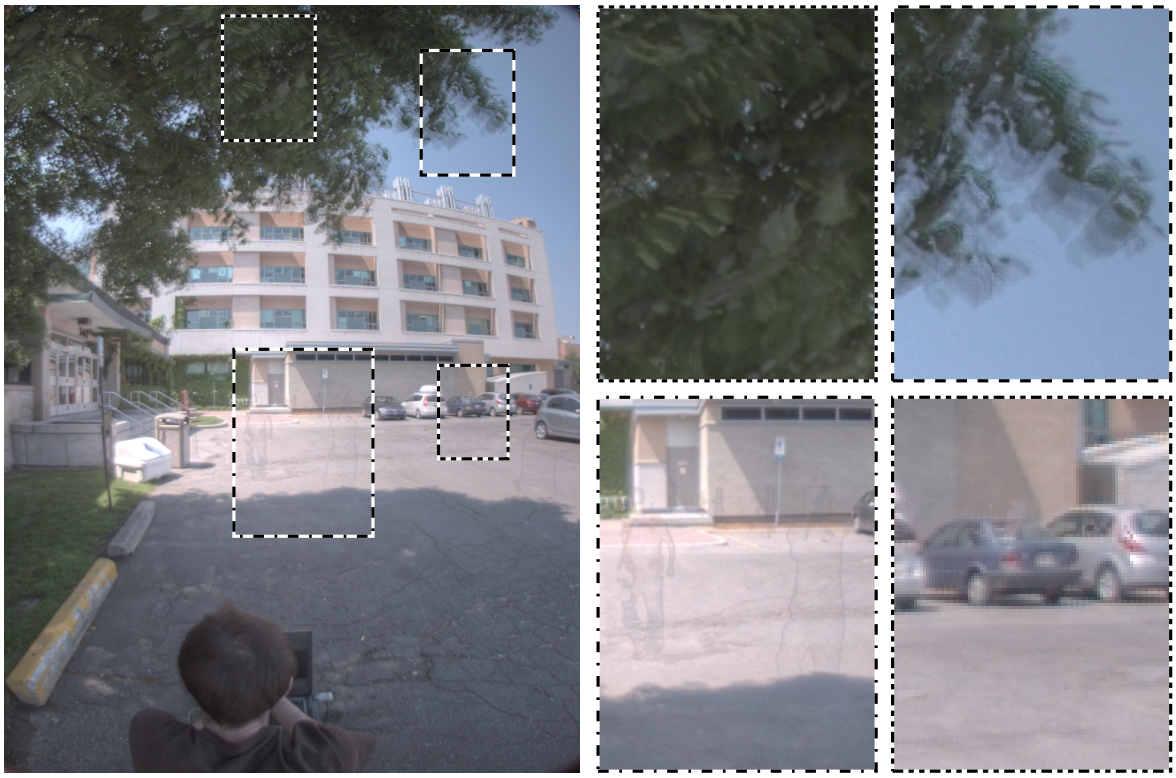


Figure 5.2: Naive HDR result for the images of Figure 5.1, with magnifications of corrupted areas. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion.

has changed by more than some threshold t_C. However, if it was poorly exposed in either image, this comparison is not valid. This concept is demonstrated in Figure 5.3, where we see that parts of the scene that were well exposed in both images take on similar values after normalization, while those that were saturated behave unpredictably, and cannot be compared even after normalization[1]. In particular, if a region was saturated in both images, it will end up with significantly different values after normalization since the saturated value is the same in both (say, 255), but the exposure time Δt by which the two images are divided is different. As a result, areas saturated in both LDR images will often trigger a naive change detection algorithm.

We make the following observations about what can be known from pixel values in consecutive *normalized* images based on their values in the LDR input images:

- If a pixel is under-exposed or over-exposed in *both* LDR images, we cannot make

[1]For illustrative purposes, we have tonemapped the normalized images.

a b

Figure 5.3: Normalization of exposure-bracketed images. a Dark exposure: original image (top) and tonemapped version of the normalized image (bottom). b Brighter exposure.

any observations about whether it changed between those images.

- If a pixel is under-exposed or over-exposed in *either* image, it does not provide reliable change detection information; to confidently perform the change detection outlined in the previous chapter, the pixel must be *well exposed* in *both* LDR images.

- Since one of the fundamental assumptions of HDR imaging is that scene illumination is constant [2], we can arrange LDR images in order of ascending exposure (brightness), in which case all pixels should be increasing in brightness up to the saturation point. Then, if a pixel gets darker, we know it has seen a moving object

in one of those images. While this change will be caught by the change detection in the case of two *well-exposed* images, this additional check will also trigger when a pixel goes from being saturated in one image to non-saturated in the next (brighter) image.

With these restrictions in mind, we can calculate several masks to aid in the change detection. In practice a mask in the image processing context is a matrix the same size as an image of interest, with ones markings areas of interest based on some criteria, and zeros elsewhere. Here we denote them as sets of pixel indices corresponding to the areas of interest. For brevity, we will use the term "saturated" to refer to pixels that are either under-exposed or over-exposed. The *areas saturated in both images* are given by:

$$\mathbf{M}_{BothSat} = \left\{(i,j)|\neg\left(t_{US} \leq z^{G}_{i,j,1} \leq t_{OS}\right), \neg\left(t_{US} \leq z^{G}_{i,j,2} \leq t_{OS}\right)\right\}, \tag{5.1}$$

where t_{US} and t_{OS} are the thresholds below and above which pixels are considered under-exposed and over-exposed, respectively, and $z^{G}_{i,j,1}$ is the grayscale value of a pixel (i,j) in image 1. Areas that have *darkened* between shots in the LDR images are given by,

$$\mathbf{M}_{Darkened} = \left\{(i,j)|z^{G,F_{avg}}_{i,j,1} - z^{G,F_{avg}}_{i,j,2} < t_D\right\}, \tag{5.2}$$

where t_D is a minimum amount by which a pixel must darken to count as having darkened. Areas in which *neither image is saturated* are given by

$$\mathbf{M}_{NeitherSat} = \left\{(i,j)|t_{US} \leq z^{G}_{i,j,1} \leq t_{OS}, t_{US} \leq z^{G}_{i,j,2} \leq t_{OS}\right\}, \tag{5.3}$$

Our naive change detection is as before except we now apply it to exposure-normalized images, $\mathbf{E}$, so the changed areas are given by

$$\mathbf{M}_{Diff} = \left\{(i,j) \left| \bigvee_{k=0}^{2} \left(\frac{\mathbf{E}^{F_{avg}}_{i,j,k,1} - \mathbf{E}^{F_{avg}}_{i,j,2}}{\mathbf{E}^{F_{avg}}_{i,j,k,1}} > t_C \right) \right.\right\}, \tag{5.4}$$

where t_C is a percentage by which a pixel's value must change between images to register as changed. With the masks outlined so far, we can now encode our overall HDR change detection logic as

$$\mathbf{M}_C = \left[(\neg\mathbf{M}_{BothSat}) \cap (\mathbf{M}_{Darkened} \cup (\mathbf{M}_{NeitherSat} \cap \mathbf{M}_{Diff}))\right], \tag{5.5}$$

which can be read as follows: to detect a pixel as having changed between two images, it cannot be saturated in both LDR images, and it must have either darkened between LDR images, or have been well exposed in both images *and* have changed by a fraction t_C between images.

Empirically, we have found the following threshold settings to work well across a large number of images:

$$\begin{aligned} t_{US} &= 1 \\ t_{OS} &= 250 \\ t_D &= -10 \\ t_C &= 0.40 \\ N_{SP} &= 1200 \end{aligned}$$

While a threshold for change detection would typically be less strict, in the range of 0.20, we found that a more strict threshold was necessary in the case of normalized exposure-bracketed images to avoid over-detection. This is likely due to inaccuracies introduced by the normalization process. Our choice of parameters is discussed in Section 6.3.

Figure 5.4 shows an example of the process outlined here, including the key masks described above, and the absolute difference image. We notice that the check for darkened areas improves detection of one of the small distant ghosts. The requirement that neither image be saturated if we are to trust the basic change detection ($\mathbf{M}_{Diff}$) means that masking of one of the nearby ghosts deteriorates somewhat, making this check look detrimental. However, this check becomes important when images are separated by large EV steps (e.g. 1EV, 2EV) as it greatly reduced erroneous change detections. The areas lost here will be recovered in the superpixel mask refinement step.

From here, the refinement of change masks by Superpixels and down-weighting of weight matrices proceeds as before with minor changes. For Superpixel refinement, since the implememtation released by Achanta et al. [77] works only on 8-bit images, we apply the segmentation to the LDR images, not the normalized images, which have floating point pixel values. For the down-weighting, we replace changed areas with the lowest non-zero weight produced by the weighting function; see Section 3.5 for details. The result of pair-wise down-weighting for the exposure-bracketed image set of Figure 5.1 is shown in Figure 5.5. As in the LDR scenarios, we see that discrete ghosts are well

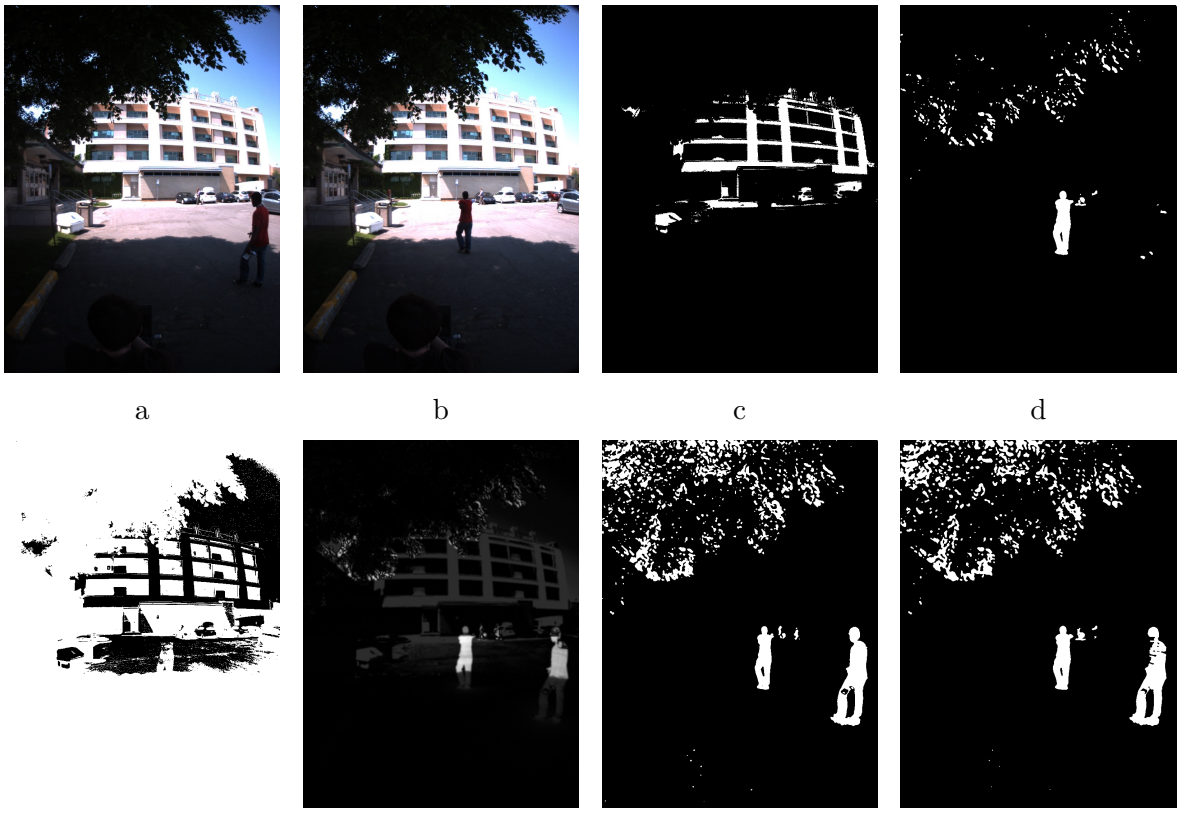


Figure 5.4: Change detection in exposure-bracketed sequences. a-b Two exposure-bracketed images, $\mathbf{Z}_1$ and $\mathbf{Z}_2$, separated by $\frac{1}{3}$EV (brighter on right). c Areas saturated in both images ($\mathbf{M}_{BothSat}$). d Areas that darkened when moving to the brighter image ($\mathbf{M}_{Darkened}$). e Areas where neither image is saturated ($\mathbf{M}_{NeitherSat}$). f Absolute Difference of the images. g Absolute difference thresholded at 40% change ($\mathbf{M}_{Diff}$). h The final mask, $\mathbf{M}_C$.

handled but fluid ghosts are still problematic and require further processing, described in the following section. Pseudocode for the overall algorithm described in this section is given in Algorithm 5.1, and details of the down-weighting and segmentation steps are given in Algorithm 5.2.

5.2 Areas Experiencing Fluid Motion

The process for replacing areas experiencing fluid motion requires several additional steps in the case of HDR deghosting. As in the LDR case, we sum the number of times all

Algorithm 5.1 Background Estimation for HDR Images - Main

Change Detection

Load N exposure-bracketed input images of size $W \times H$
Load exposure info (Exposure time and Gain) for each image
if Response is not linear **then**
    Convert to irradiance values using recovered response curve $g(\cdot)$
end if
Normalize irradiance image ▷ Divide each image by its exposure time and gain
Create N weight matrices $\mathbf{W}$, sized $W \times H$, initiliazed using HDR weighting function
$\mathbf{M}_A = \{\}$ ▷ Mask to track what areas see change throughout frames
for Each pair of consecutive images, $\mathbf{E}_1$ and $\mathbf{E}_2$ **do**
    Calculate $\mathbf{M}_{BothSat}$, $\mathbf{M}_{Darkened}$, $\mathbf{M}_{NeitherSat}$, $\mathbf{M}_{Diff}$, and $\mathbf{M}_C$ (see Expressions
    5.1-5.5).
    Segmentation to get $\mathbf{M}_{CR}$ (see Algorithm 5.2)
    Down-weighting (see Algorithm 5.2)
    $\mathbf{M}_A = \mathbf{M}_A \cup \mathbf{M}_{CR}$
end for
Compose HDR background image, $\mathbf{H}$ using modified weights
Fluid Motion Fix (see Algorithm 5.3)

Algorithm 5.2 Segmentation and Down-Weighting Routines

Segmentation

$\mathbf{M_{CR}} = \{\}$ ▷ Refined change mask for current image pair
for Each of $\mathbf{E}_1$, $\mathbf{E}_2$ **do**
    Run Superpixel segmentation on corresponding $\mathbf{Z}$ to get $\mathbf{S}$
    $\mathbf{M_{CR}} = \mathbf{M_{CR}} \cup \left\{ \mathbf{S}_p \,\middle|\, \frac{|\mathbf{S}_p \cap \mathbf{M}_C|}{|\mathbf{S}_p|} > t_{SParea} \right\}$
end for

Down-weighting

for Each of $\mathbf{E}_1$, $\mathbf{E}_2$ **do**
    Reduce weight matrix for $\mathbf{E}$ where marked by $\mathbf{M}_{CR}$
end for
Feather edges of newly down-weighted areas

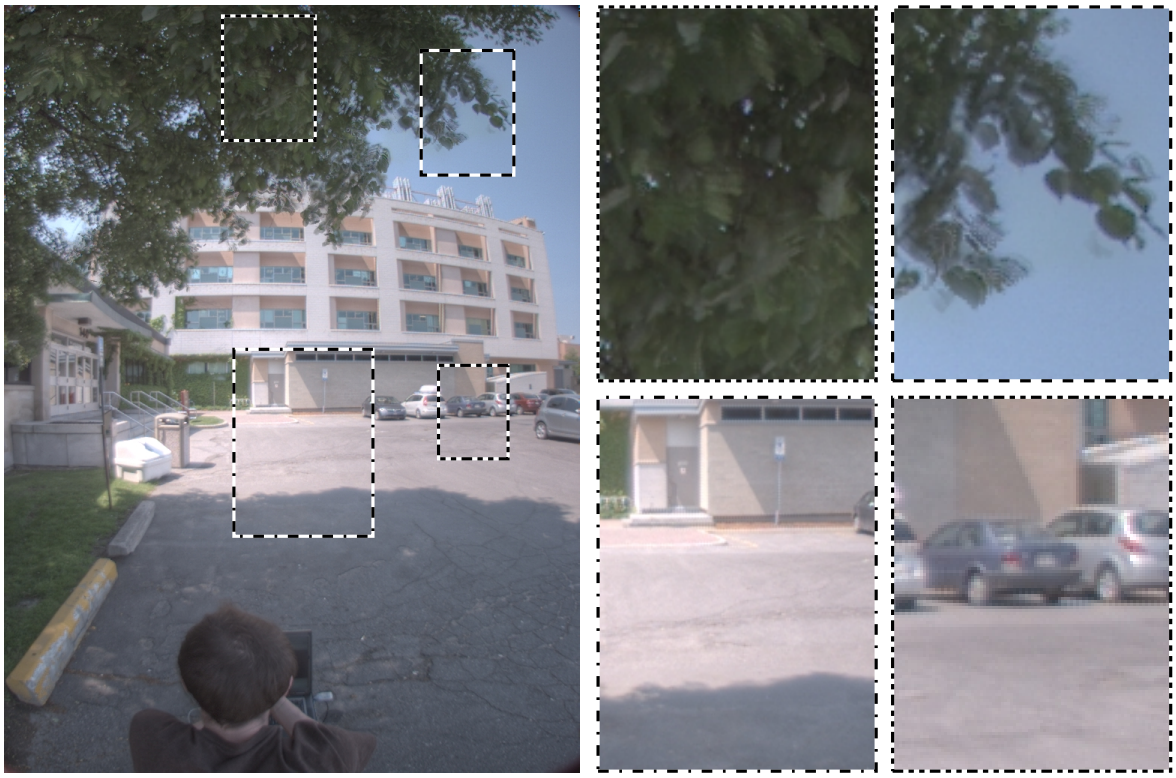


Figure 5.5: The result of pair-wise down-weighting alone for the images of Figure 5.1, with magnifications of corrupted areas. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion. Note that discrete ghosts are virtually eliminated, but fluid ghosts are only somewhat improved.

pixels in an image have seen a change to get $\mathbf{S}_{MovingCount}$. In the LDR version, we count a pixel as always moving if it moved in more than some threshold fraction, t_{MC}, of the total number of images, N, e.g. if it moved in at least 25% of the input images. In the HDR version, images in which a given pixel is saturated should not count towards the total number of images for that pixel, so we now threshold based on a fraction of the total number of times a pixel was well exposed, given by:

$$\mathbf{S}_{WellExposed} = \left\{ s_{i,j} \mid s_{i,j} = \left| \left\{ n \mid t_{US} < z^G_{i,j,n} < t_{OS} \right\} \right| \right\}, \tag{5.6}$$

where an evaluation of $\left| \left\{ n \mid t_{US} < z^G_{i,j,n} < t_{OS} \right\} \right|$ simply returns the number of times pixel (i, j) was well exposed. Now, we find fluid motion pixels as

$$\mathbf{M}_{FluidMotion} = \left\{ (i,j) \left| \frac{\mathbf{S}_{MovingCount}(i,j)}{\mathbf{S}_{WellExposed}(i,j)} > t_{MC} \right. \right\}. \tag{5.7}$$

As before, we now break this mask into regions using connected component labeling, and for each of these regions we seek the best source image from the input image stack to replace it. In the LDR version, we searched for the replacement image best matching the background for the current region, where the background was approximated by the naive combination (direct average) of the input images. Since we are dealing with exposure-bracketed images, we take inspiration from previous deghosting methods [30, 31] and now look for the image with the best exposure in the current region. For each input image, we calculate the sum of the HDR weights in the current region; the image with the highest sum is chosen as a replacement for this region. In the LDR version, since all images are identically exposed, once the best input image is found, we can naively pull the entire region from it. This approach is also used in some of the existing approaches to HDR ghost removal [30, 31]. However, if the fluid motion area happens to be due to a dark object moving against a bright background, this can result in artifacts if we take oversaturated pixels from the best-exposed image, as shown in the upper-left zoom-box of Figure 5.6. The gray areas appear because they were saturated in the selected input image. They appear grayscale here because they were fully saturated, i.e. RGB $= (255, 255, 255)$, and they are gray rather than white because the chosen replacement image was one of the brighter input images. In brighter images, a saturated value is divided by a smaller exposure time and therefore ends up dark after normalization when compared to areas that were saturated in darker images. This effect is not seen in the naive result because these saturated areas are given zero weights in a weighted average.

Therefore, when replacing the current region with information from the best exposure, we reject any pixels whose intensity is below or above the new thresholds t_{MFUS} and t_{MFOS}, respectively. In our current experiments we have achieved good results with $t_{MFUS} = t_{US}$ and setting t_{MFOS} to 90% of Z_{max}, i.e. 230 for 8-bit images. The reason for decreasing t_{MFOS} is that we are replacing pixels in the final HDR image with information from a single input image, and therefore, we want to pull those pixels from images in which they were well within the accurate portion of the camera response curve. Early testing revealed that $t_{MFOS} = t_{OS}$ was insufficient and still resulted in obvious gray artifacts. The result of simply rejecting saturated pixels from the best replacement image is shown in Figure 5.7, which shows that this fix alone improves over pair-wise down-weighting, while avoiding the gray artifacts resulting from naively accepting the information in the chosen input image.

However, we can improve further upon this result by noting that pixels which were

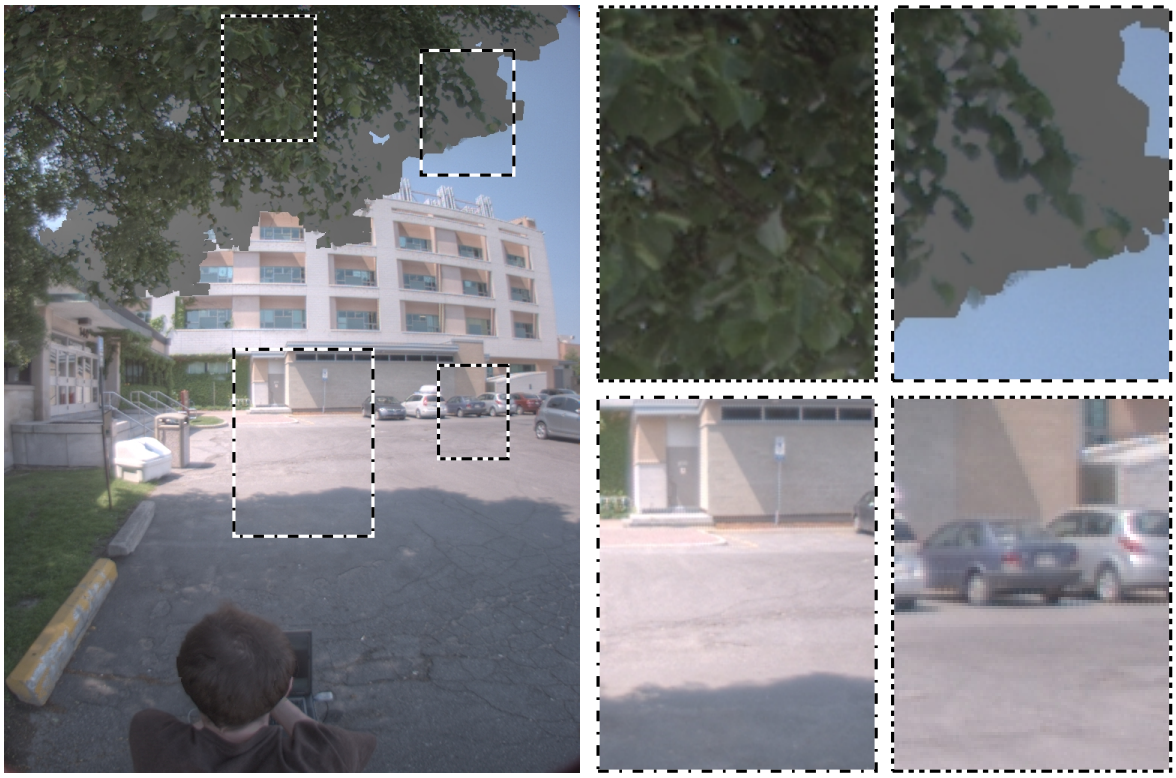


Figure 5.6: Fluid motion fix: pulling entire fluid motion area from one input image with no saturation check. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion. Note that the top-left area is greatly improved, but the top-right area contains gray artifacts in areas that were saturated in the best exposure.

saturated in the chosen input image may be well-exposed in some other input image. We explored two approaches to further improve the replacement of fluid motion regions by iteratively replacing regions that are saturated in the best input image.

In the first approach, for pixels that are *over-saturated* in the current image, we move to the next-brightest image (i.e. one step darker). Such a pixel can be no darker in the new LDR image than its previous value divided by the relative EV change. For example, if we start in an image with an exposure time of $\frac{1}{2}$s, and move to a darker image with an exposure time of $\frac{1}{4}$s, a pixel with a value of 255 in the first image should be no darker than $255 \times \frac{1/4}{1/2} = 127.5$ in the second image. So the approach starts by finding the overall best-exposed image for the fluid motion region then replacing that region in the combined image $\mathbf{H}$ with information from the best exposure, except where the best-exposed image is over-saturated. In these over-saturated regions, we look at the next-brightest image and replace these regions with information from that image except where it is *also* over-

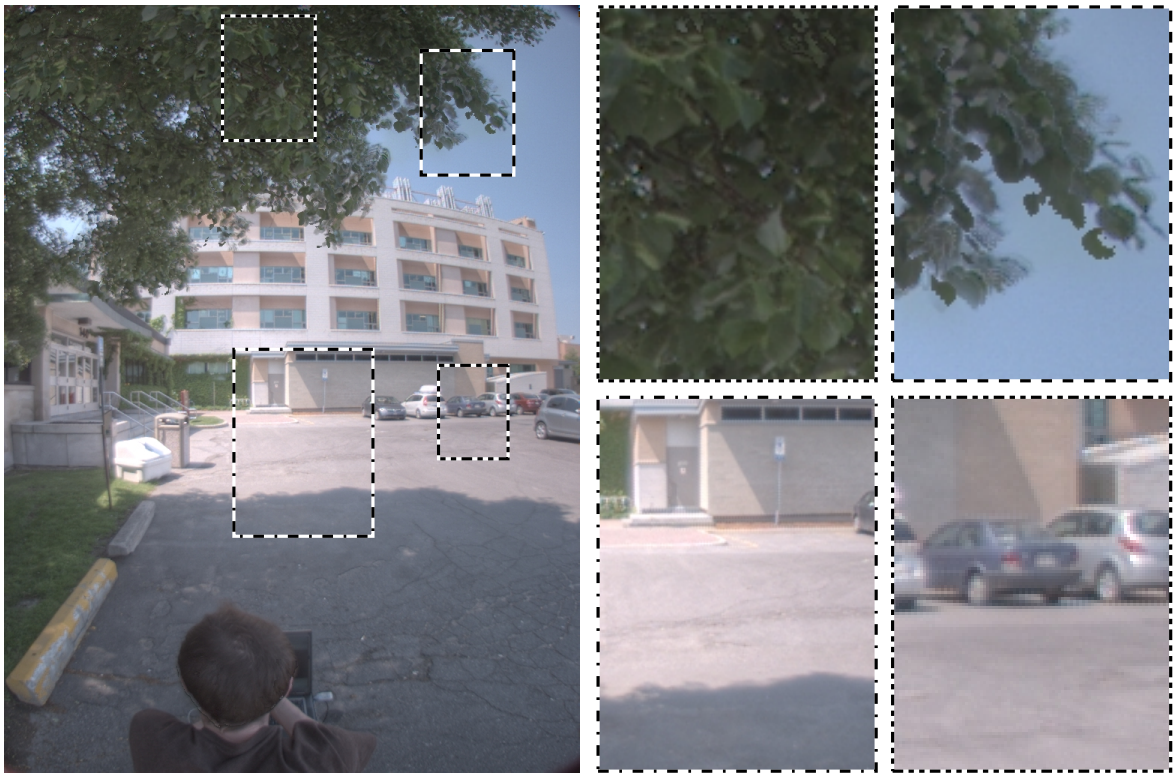


Figure 5.7: Fluid area fix: pulling entire fluid motion area from one input image except where that image is saturated. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion. Note that the saturation-related artifacts of Figure 5.6 are gone, but ghosts near saturated regions have returned to a combination of the naive and deghosted results.

saturated or fails the relative-exposure check. This process continues for the current fluid motion region until the region has been fully replaced or we run out of images. If, after checking all input images, some pixels have not been replaced, these are pulled from the naive HDR result. A similar process is repeated for *under-saturated* portions, where we proceed to brighter images and ensure that the new value divided by the relative EV is below t_{MFUS}. The result of this fluid motion fix is shown in Figure 5.8. We can see that is has improved sharpness in the borders of the foliage somewhat, but is still imperfect.

In the second approach, for pixels that could not be replaced with the current best exposure due to saturation, we look for a new best exposure. This means that rather than step through images sequentially, we may skip several images at a time in the search for a new best exposure. This process iterates until no pixels remain or we have exhausted the input images. If any pixels in the current region cannot be replaced from any input

images, they are again taken from the naive HDR result as a fallback. Here, the process is the same for both under- and over-saturated regions. If we think of this process as labeling pixels with the indices of the replacement image, then in practice, this second approach tends to produce a smoother labeling in the fluid motion regions, which is one of the goals of the graph-cut methods discussed in Section 2.3.3. The result of this fix is shown in Figure 5.9, which is clearly a large improvement over the naive and pair-wise down-weighting results.

Figure 5.10 shows the same image with blending applied around the edges of replaced regions. There are still some trace ghosting artifacts around the boundaries of the replaced areas, in some cases manifesting as small parts of leaves that are not attached to the main body of foliage. However, when viewed at the full-image level, the reduction in ghosting is effectively complete. Pseudocode for the second version of our fluid motion fix is given in Algorithm 5.3.

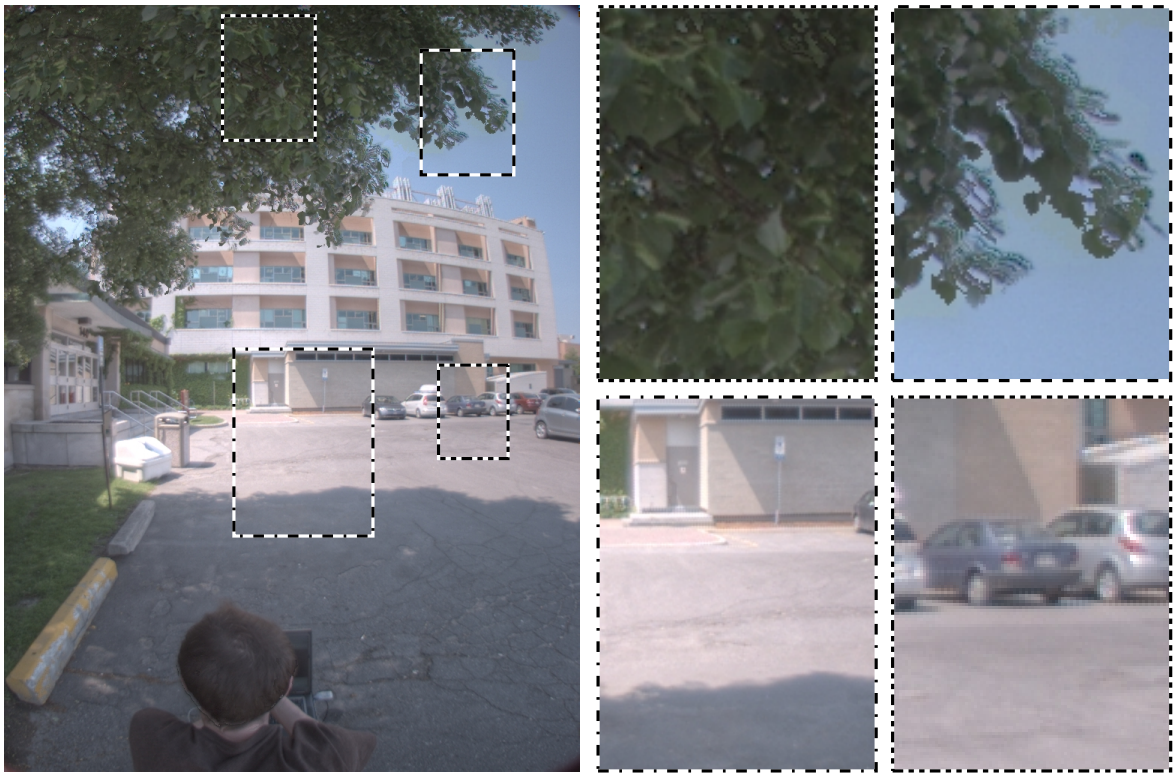


Figure 5.8: Fluid area fix: replacement of saturated areas in current exposure with information from next brightest image. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion. Note that areas around the edges of fluid motion regions are more crisp, though we see in the top right magnification that some artifacts remain.

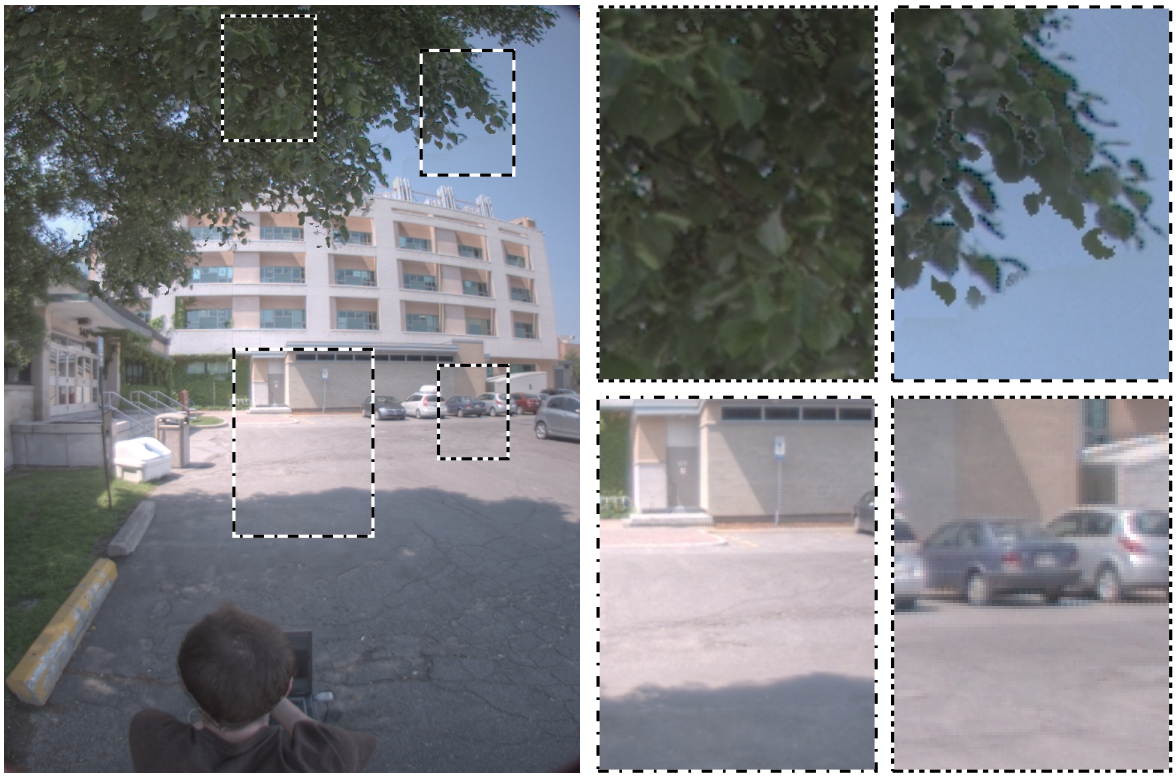


Figure 5.9: Fluid area fix: iterative replacement of saturated areas in current best exposure with information from new best exposure for those areas. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion. Note that overall result shows eliminated discrete ghosting and significant reduction of fluid ghosting.

5.3 Summary

In this chapter we have extended the two-part LDR background estimation of the previous chapter to perform ghost removal for HDR image compositing. We have outlined checks required to apply change detection approaches to exposure-bracketed images, and additional steps required in the fluid motion region fix. We have shown the success of our approach on a challenging ghosting scenario containing a mix of discrete ghosts, fluid ghosts, and backlighting on fluid ghosts which makes them additionally challenging. Pseudocode for the approach introduced in this chapter is presented in Algorithms 5.1–5.3. In Chapter 6, we present further results and compare our method to existing approaches. The effect of varying the parameters introduced in this chapter and the last will be discussed in Section 6.3.

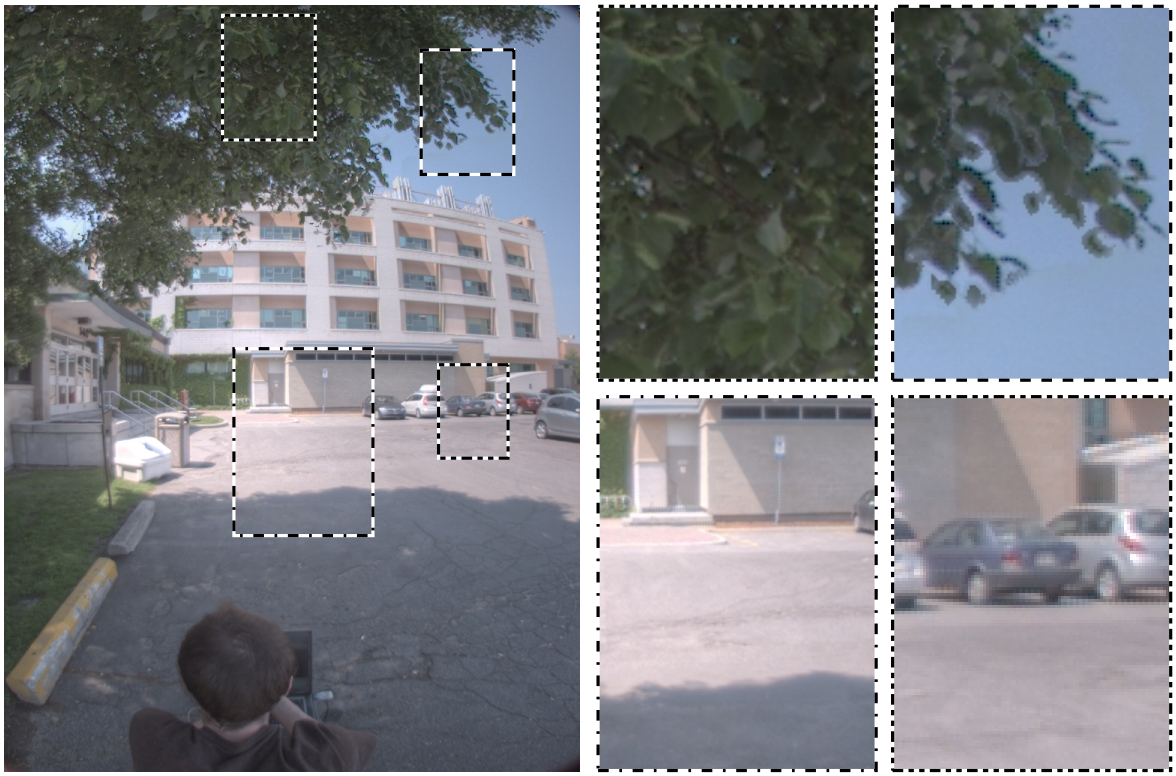


Figure 5.10: Same fluid area fix as Figure 5.9, but with blending around replaced areas. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion. Note that overall result shows eliminated discrete ghosting and significant reduction of fluid ghosting.

Algorithm 5.3 Fluid Motion Fix Routine

Fluid Motion Fix

Erode $\mathbf{M}_A$ and do connected component labeling to get $\mathbf{M}_{AE} = \{\mathbf{M}_{AE_1}, \ldots, \mathbf{M}_{AE_R}\}$
Sum all inter-frame change masks to get matrix $\mathbf{S}_{MovingCount}$
Sum number of times each pixel was well explosed to get $\mathbf{S}_{WellExposed}$
$\mathbf{M}_{FluidMotion} = \left\{(i,j) \left| \frac{\mathbf{S}_{MovingCount}(i,j)}{\mathbf{S}_{WellExposed}(i,j)} > t_{MC} \right.\right\}$
$\mathbf{M}_{FluidMotionRegions} = \left\{\mathbf{M}_{AE_r} \left| \frac{|\mathbf{M}_{AE_r} \cap \mathbf{M}_{FluidMotion}|}{|\mathbf{M}_{AE_r}|} > t_{MFA} \right.\right\}$
for all Regions $\mathbf{M}_{CurRegion}$ in $\mathbf{M}_{FluidMotionRegions}$ **do**
    while $|\mathbf{M}_{CurRegion}| > 0$ and haven't tried all images yet **do**
        $n_{BestExp} = \arg\max_n \left(\sum_{i,j} \mathbf{W}_n(i,j)\right), \quad (i,j) \in \mathbf{M}_{CurRegion}$
        Replace $\mathbf{M}_{CurRegion}$ in $\mathbf{H}$ with data from $\mathbf{E}_{n_{BestExp}}$
        Feather edges of replaced region
        $\mathbf{M}_{BestImgWellExposed} = \left\{(i,j) \left| t_{MFUS} < \mathbf{Z}_{i,j,n_{BestExp}} < t_{MFOS} \right.\right\}$
        $\mathbf{M}_{CurRegion} = \mathbf{M}_{CurRegion} \cap \neg \mathbf{M}_{BestImgWellExposed}$
    end while
    if $|\mathbf{M}_{CurRegion}| > 0$ **then**
        Replace current region with values from naive HDR result.
    end if
end for

Chapter 6

Results and Comparisons

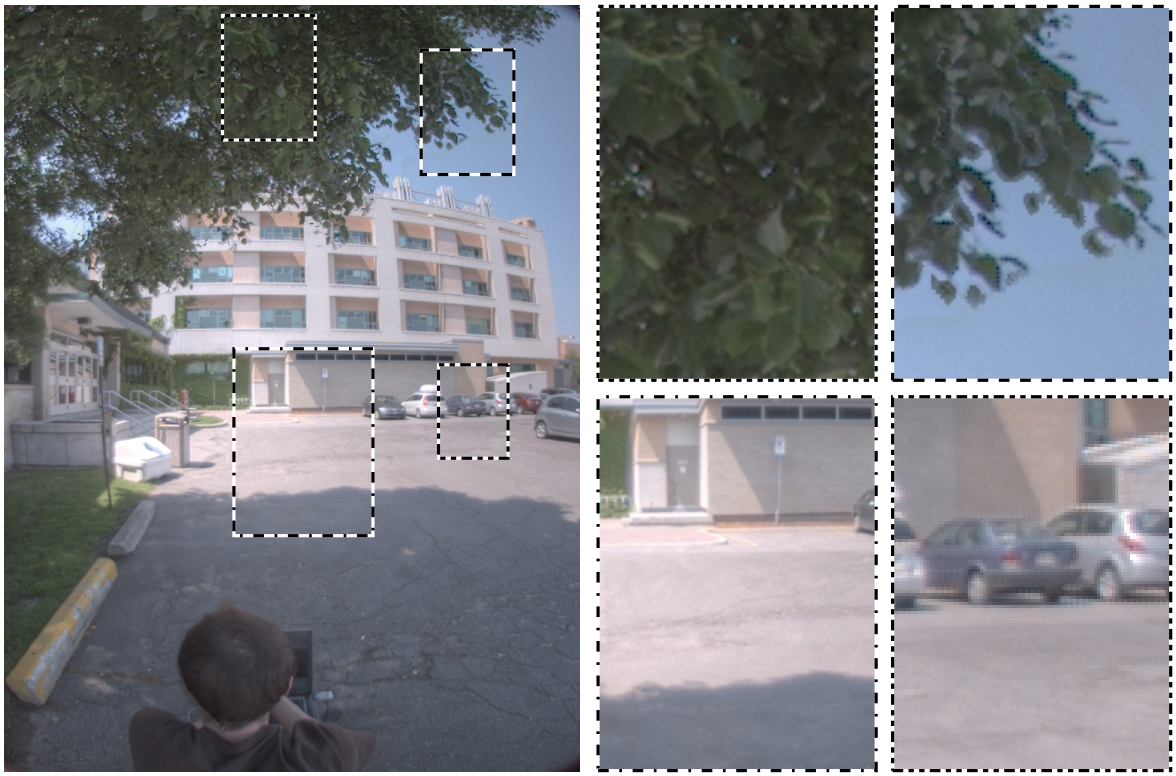


Figure 6.1: Result of the proposed method for the Marion Lot data set of Figure 5.1, with magnifications of motion-corrupted areas. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion.

In this chapter we review our results from Chapter 5 and other data sets. For several data sets, we provide comparisons with results from state of the art existing approaches

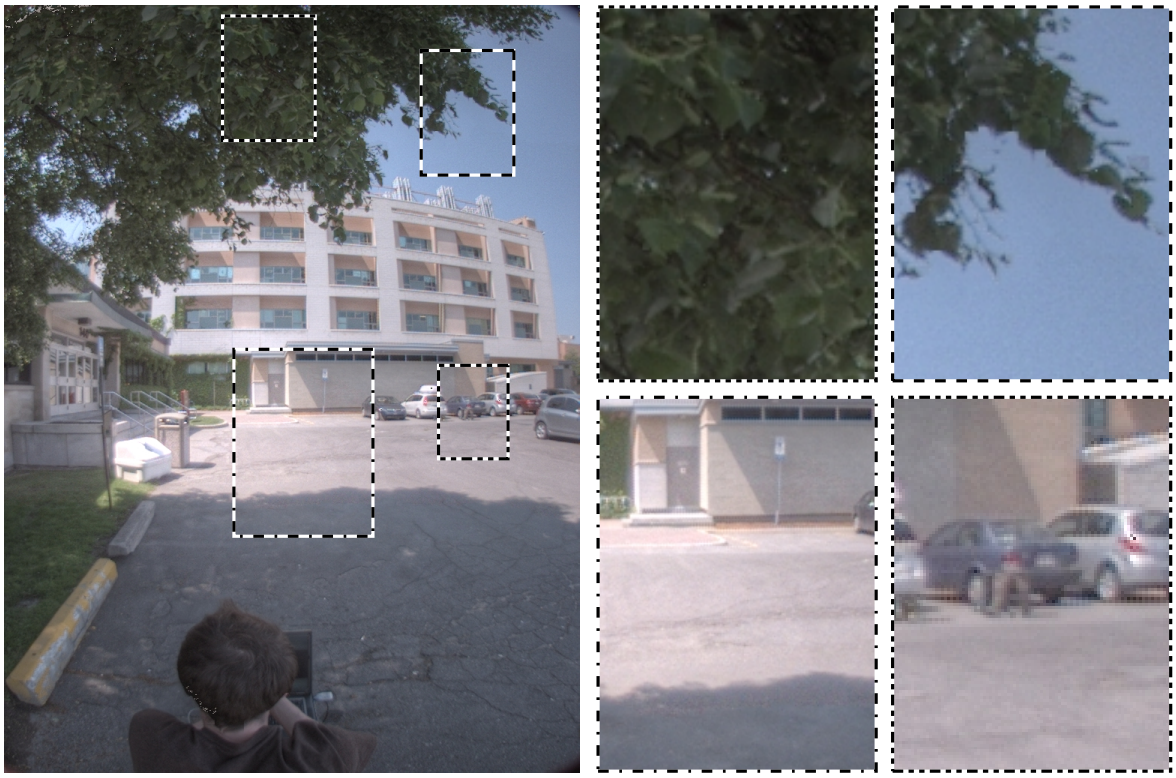


Figure 6.2: The result of Granados et al. [33] for the Marion Lot set with magnifications of corrupted areas. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion.

from Granados et al. [33] who provided their code for testing, Zimmer et al. [29] who processed several data sets for us, and from the leading commercial HDR software package Photomatix Pro. We also tested the method of Khan et al. [40], which is implemented in the open source panorama software Hugin[1]. However, we found it provided virtually no improvement for the images tested when compared to the naive result, and it is unclear whether this is from limitations to the approach or just an implementation bug in Hugin so we withhold further comment on that method.

6.1 Results and Visual Comparison

Here we provide a discussion of visible differences in the results of our methods as compared to state of the art methods and commercial software. We do not provide a quanti-

[1] http://hugin.sourceforge.net/

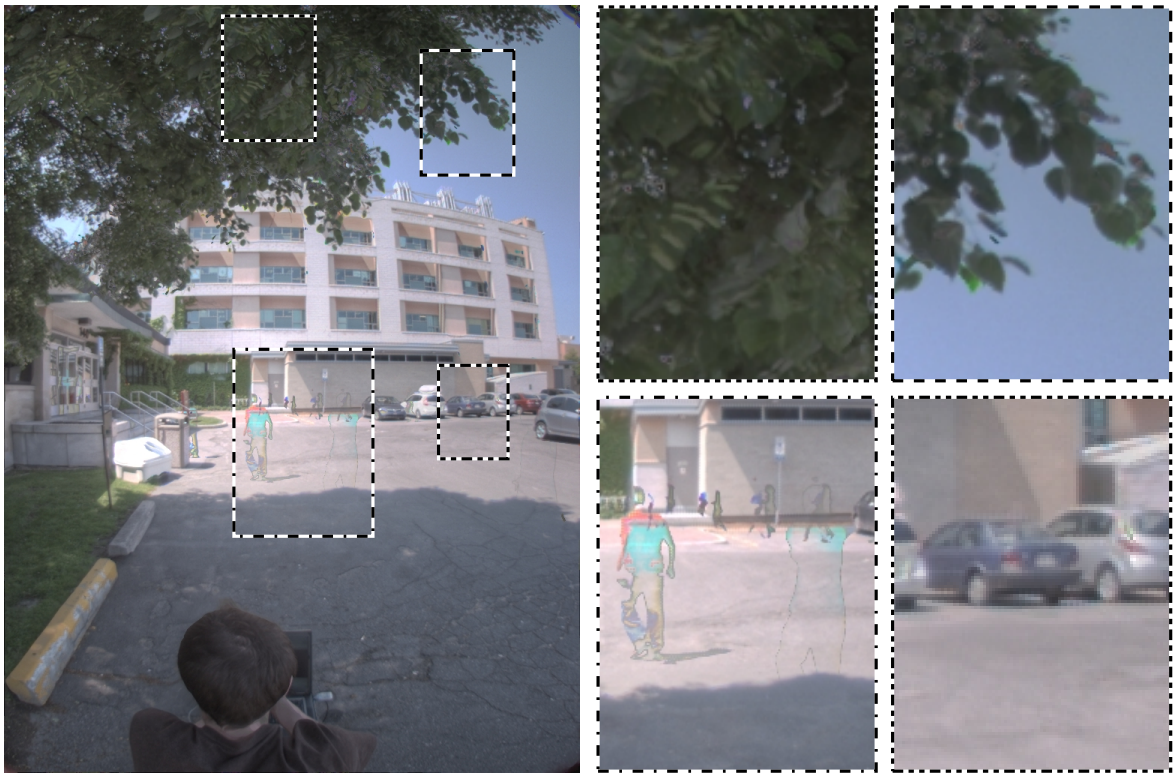


Figure 6.3: The result of Zimmer et al. [29] for the Marion Lot set with magnifications of motion-corrupted areas. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion. Note that the authors state that their method cannot fix large ghosts who displace by more than their own size between frames.

tative comparison since we are not aware of an established method for comparing in the case of fluid ghosts which, as discussed in Section 4.4, often have multiple equally valid solutions. See Section 6.3.2 for more detail on the challenges of quantitative analysis.

Figure 6.1 shows our result again on the Marion Lot data set introduced in Chapter 5, where the top two zoom boxes experienced fluid motion and the bottom two discrete motion. The LDR input images and naive result were presented in Figures 5.1 and 5.2 respectively. Figure 6.2 shows the result of Granados et al. [33] on the same set. At a macro level, the two results are quite similar. Under close inspection we see that the bottom-left zoom box, which saw two large discrete ghosts in the form of pedestrians walking rapidly, is handled identically by both methods. The lower-right area, which saw smaller discrete ghosts, shows remnants of ghosts with Granados' method in the form of portions of pedestrians. In that area, we see a pair of detached legs in front of the dark coloured car, and feet near the bottom of the lighter coloured car on the left.

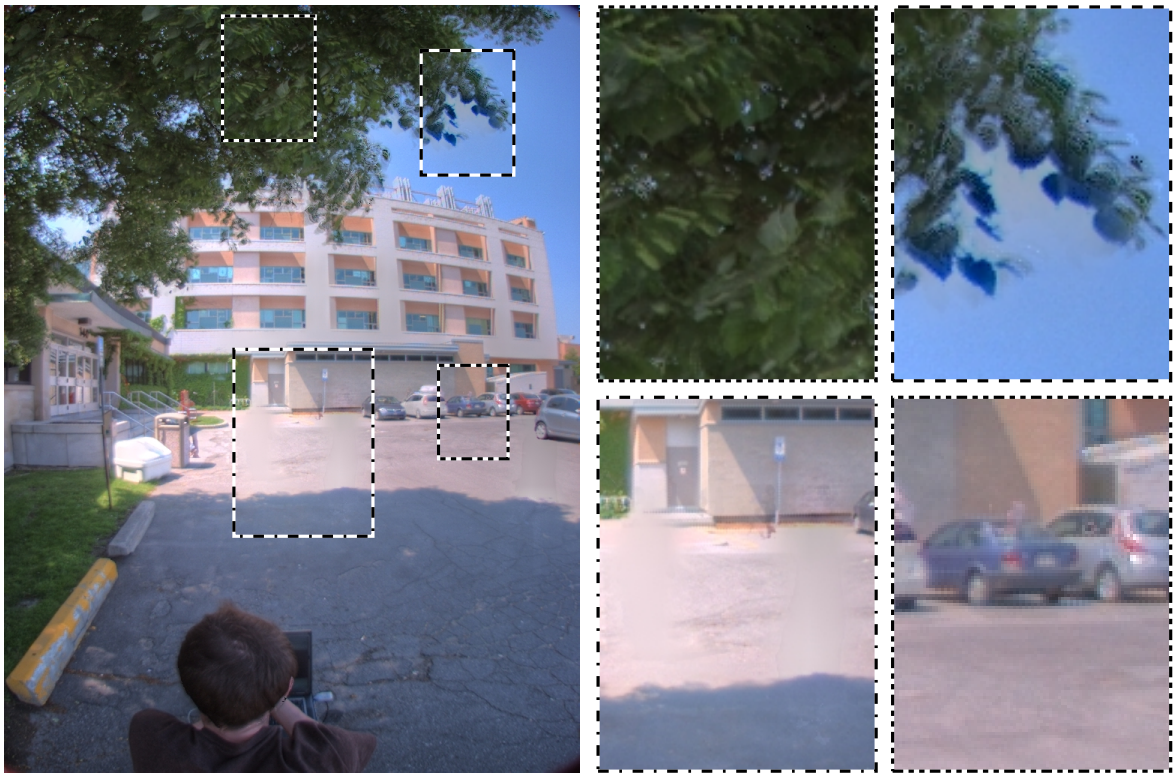


Figure 6.4: The result using Photomatix for the Marion Lot set with magnifications of motion-corrupted areas. The top two magnified areas experienced fluid motion, and the bottom two experienced discrete motion. Note in upper-left zoom box that the interior of foliage areas are only slightly improved over naive result, and in lower-left zoom box that large discrete ghosts appear to have been removed by a blurring effect, with a negative effect on the final result.

In the fluid motion areas, we see that both methods perform similarly in the center of the foliage (upper left zoom box). We note here that the two methods have clearly chosen different input image for these areas but, as in Section 4.4, we argue that for many fluid motion areas any visually plausible replacement is valid, so the two methods can be considered to have performed equally well here. The result in the upper-left zoom box is quite different between the two methods. Since the method of Granados et al. encourages choosing values for a pixel that are seen more often, in the case of branches waiving in the wind, it will often have a "pruning" effect which reduces the overall area of the branch and its leaves. This effect can be understood by considering the same zoom box for the naive result in Figure 5.2 where a medium-sized bundle of leaves waiving has created a large ghosting area. The areas which are completely blue saw only sky, and those that are completely green saw only leaves. Areas in between saw a combination

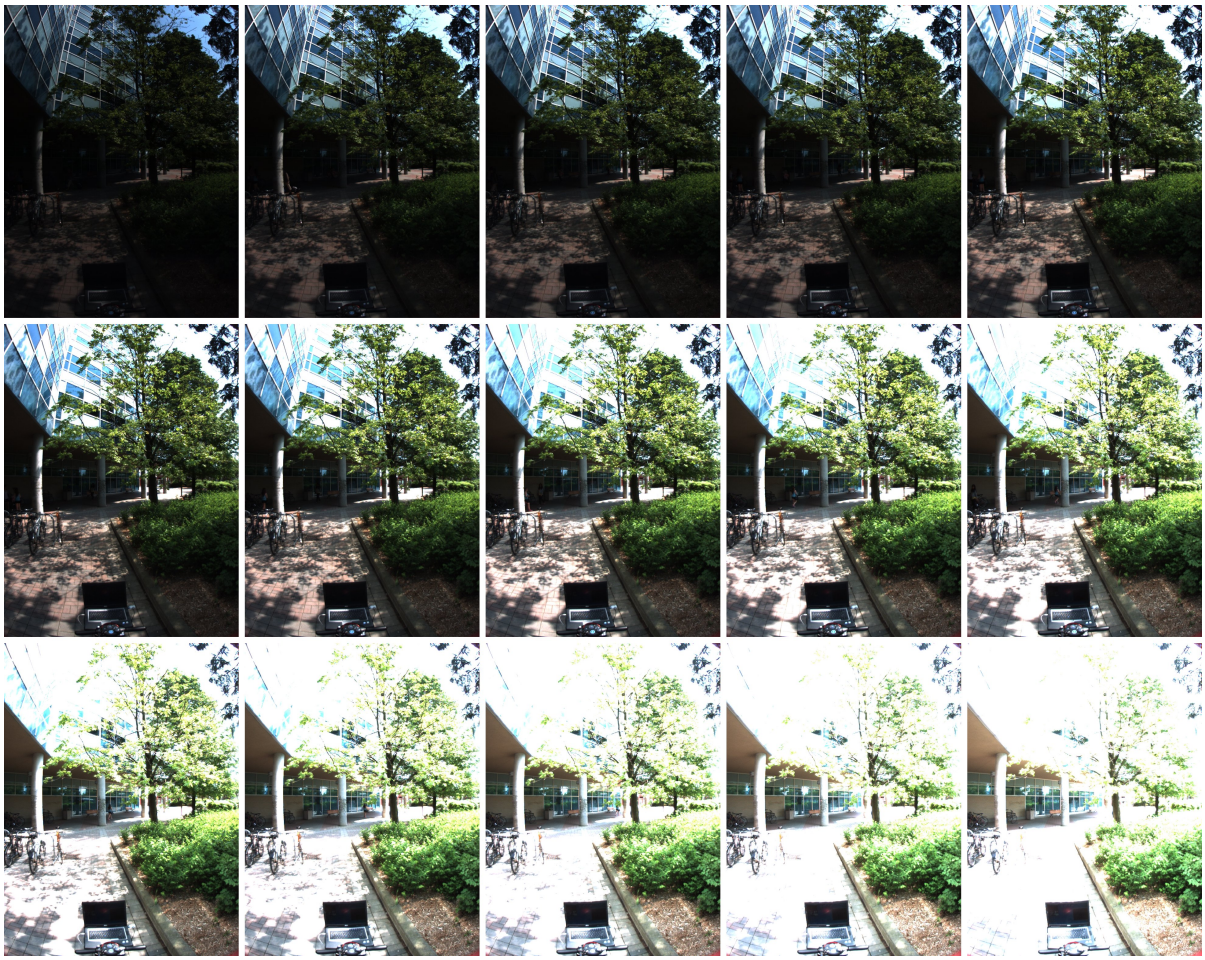


Figure 6.5: Careg Courtyard data set: 15 out of a 16-image sequence of exposure-bracketed images of the same scene displaying motion corruption. The scene is corrupted by multiple people walking through (discrete ghosts), as well as a several sections of foliage blowing in the wind (fluid ghosting).

of both, with the ratio of blue to green indicating how often one or the other was seen. In any given frame, the branch would have a medium size, but for pixels that saw both sky and leaves, and saw sky more often than leaves, the energy term of Granados et al. penalizes keeping the leaves. The result is a pruning effect, with the end result being a reduction in size of the branch when compared to any one of the input images. However, when viewed alone, the result is still visually plausible. In our result, some small black artifacts are visible around the edges of the leaves under high magnification; these are demosaicking artifacts present in the source images. We also see some gray fringing around the edges of some leaves; this is a result of the blending in the proposed method.

Figure 6.6: The naive result for the Careg Courtyard data set, with magnifications of motion-corrupted areas.

On the topic of blending, we make the following notes:

- Discontinuities will often occur at locations where we switch from taking values from one input image to another ("labeling transitions" in graph cuts terminology).
- We currently blend using a simple center-weighted average [111, Sec.6.2], which generally improves results in the vicinity of labeling transitions.
- Our method would benefit from a more advanced approach such as Poisson blending, which is used in the approaches of Granados et al. [33] and Gallo et al. [32].
- Existing methods [30–34] have shown labeling transitions to be a challenge, so generally it would be good to avoid them.
- Our method strategically starts with pair-wise down-weighting rather than jumping straight to replacing all motion-corrupted regions with single input images. This

Figure 6.7: Result of the proposed method for the Careg Courtyard data set, with magnifications of motion-corrupted areas.

preemptively eliminates most labeling transitions, thereby reducing the number of regions requiring blending.

Figure 6.3 shows the results of Zimmer et al. on the same data. Note that the authors do not claim to be able to address discrete ghosting, so the only point of comparison here is the foliage, and even there, the authors note that this is a particularly challenging scenario [112]. Their result in the center of the foliage shows an improvement over the naive result, but ghosting clearly remains and manifests as doubling of some of the leaves. Their result around the edges of the foliage in the upper-right zoom box are very good.

Finally, Figure 6.4 shows the deghosting result using the software package Photomatix Pro from HDRsoft. Note that Photomatix adjusts colour balance and saturation during the HDR combination process, the effect of which is clearly visible despite all results for this set being tonemapped with the same tonemapping operator. Areas within the foliage appear similar to the naive result, while those at the edges are improved over the naive result, but not as good as the proposed method or the others compared here. Discrete

Figure 6.8: The result of Granados et al. for the Careg Courtyard data set, with magnifications of motion-corrupted areas.

ghosts are diminished, though this appears to be partly accomplished by blurring areas that experienced ghosting, which is particularly noticeable in the bottom-left zoom box.

Figure 6.5 presents the Careg Courtyard data set, and the naive result for this set is shown in Figure 6.6. The top zoom box shows discrete motion due to pedestrians, and the lower shows fluid motion due to foliage which, unlike the Marion Lot data set, is not sharply backlit.

Figures 6.7-6.9 show the results of the proposed method, Granados et al., and Photomatix. We see that the proposed method has fully eliminated the discrete ghosts, and has sharpened a portion of the foliage. The result of Granados et al. has similarly removed most of the discrete ghosts, though it has kept a small portion of a pedestrian just to the right of the left-most concrete column. They have also sharpened a greater fraction of the waving foliage. The Photomatix result has struggled with the discrete ghosts in this scenario, removing only two of them, and has also failed to significantly improve the blurred foliage.

Figures 6.10-6.15 present further results of our method, showing improvement of

Figure 6.9: The result using Photomatix for the Careg Courtyard data set, with magnifications of motion-corrupted areas.

both discrete and fluid ghosts. In place of a detailed discussion of these results, we have highlighted areas of interest in each result, we provide a brief discussion under the figures when necessary, and we provide the naive result for comparison.

6.2 Speed Comparison

Here, we offer a brief informal discsussion on speed. A formal comparison is not possible at this time because we do not have access to the code for all methods, and they have been implemented in different languages, optimized to varying degrees, and run on different hardware. For the data sets shown so far, the proposed deghosting method was implemented in MATLAB. It uses the C++ implementation of SLIC superpixels via a MEX (MATLAB Executable) wrapper, but has not otherwise been optimized. It and Photomatix were both executed on a Dell Inspiron 1564 with an Intel Core i3-330m processor and 4GB RAM, running 64-bit Windows 7. Code for the method of Granados et

al., provided by the authors, is implemented in C++ and run in a Linux virtual machine with the same processor and 2GB RAM. The method of Zimmer et al. was executed using a Pentium 4 3.2GHz, and the results shown were produced with their non-optimized C version rather than their optimized GPU version, and without the Super-Resolution component of their method.

The proposed method had typical runtimes between 112 and 180 seconds. The method of Granados et al. takes 15-20 minutes. For a test set of our images, the method of Zimmer et al. had runtimes ranging from 7.5 to 50 minutes as the multi-scale pyramid downsampling factor ranged from 0.5 to 0.95. The result shown in Figure 6.3 uses a value of 0.95 for this parameter and took 50 minutes to complete. Photomatix takes several seconds to complete, making it the fastest method by far, but as shown in Figures 6.4 and 6.9, its results are also far inferior to the other methods. We predict that an optimized C++ version of our proposed method would operate considerably faster than it currently does, possibly at speeds comparable to Photomatix.

Figure 6.16 shows a histogram of execution times for the proposed method based on 228 images sets, each consisting of 16 images of resolution 1024x768. 87% of tests completed in under 3 minutes.

6.3 Parameters

In Chapters 4 and 5, we introduced multiple parameters. So far these have been set experimentally in an attempt to find single settings for each one that provide good results across a broad range of input data sets. We provide here a brief discussion on and demonstration of the effects of varying the parameters, then briefly discuss the challenges in performing a rigorous ground truth-based optimization.

6.3.1 Parameter Values

The effects of parameters t_C, N_{SP}, t_{SParea}, and t_{MC} are demonstrated and discussed in Figures 6.17, 6.18, 6.19, and 6.20 respectively. Our reference settings are $t_C = 0.40$, $N_{SP} = 1200$, $t_{SParea} = 0.25$, and $t_{MC} = 0.25$, and in each demonstration only one parameter is varied. The effects of t_C, N_{SP}, and t_{SParea} can be directly seen and assessed in the pair-wise down-weighting result (before the fluid motion fix is applied), so for these parameters we only consider their effects prior to the fluid motion fix.

The first key parameter is the basic change detection threshold, t_C, which determines

by what percentage a pixel's value must change between images to count it as motion-corrupted. Lowering t_C to some trivial low value, say 1%, results in detecting more changes, which is generally good if the system is noise-free and saturation-free. In reality, noise and saturation effects mean that too low a threshold results in over-detection. In the case of exposure-bracketed images, we have found that as we progress through increasing exposures, when regions transition from well-exposed to saturated, they are likely to trigger with a low t_C. If we consider a particular region of the input images, if t_C is low enough that the region is detected as changed in every frame, the result is that every input gets equally small weights, reverting the result for that area to the naive result. Therefore, t_C needs to be increased from the trivially low point to a more restrictive value. Increasing it too far, however, will result in genuine changes being missed, with the result that ghosts will slip through again.

The next parameter of interest is the number of superpixels, N_{SP} used in the refinement of the change detection mask $\mathbf{M}_C$. Increasing N_{SP} reduces superpixels' average area. When they are very large, it is rare for $\mathbf{M}_C$ to mask enough of a given superpixel to mark the whole superpixel as changed, so portions of large ghosts or entire small ghosts can be missed. As N_{SP} is increased, we generally capture ghosts more completely. However, if N_{SP} is increased too far, it becomes possible for small portions of large ghosts to be missed because $\mathbf{M}_C$ typically misses some of the ghosts, and if the superpixels are small enough, they can exist entirely within a portion of a ghost that has been missed in $\mathbf{M}_C$. Our experiments have worked well with $N_{SP} = 1200$. Another option would be to segment the image at multiple superpixel resolutions, using each to refine $\mathbf{M}_C$ separately and taking the union of the results. We have tested this approach and in our current implementation the time penalty is quite high, though it does appear to slightly improve some results by taking the guesswork out of setting N_{SP}. In an optimized implementation, it should be possible to add this approach with minimal time penalty.

Related to N_{SP} is the paramater t_{SParea}, which dictates what percentage of a superpixel's area must be marked by $\mathbf{M}_C$ in order to count the entire superpixel as moving. Recalling that the purpose of the superpixel selection is to refine $\mathbf{M}_C$ by filling in gaps and growing $\mathbf{M}_C$ somewhat, it is easy to see that the maximum value for t_{SParea}, 100%, makes it very hard to select superpixels as moving, resulting in a degredation of $\mathbf{M}_C$. Lowering N_{SP} from there will result in more superpixels being selected. Lowering it to an extreme point, say to 1%, will result in many superpixels being detected unecessarily since a smattering of noise in $\mathbf{M}_C$ within a superpixel's area will be enough to count it. Like setting t_C too low, this can result in some areas being unnecessarily or repeatedly

counted as moving, resulting in a fall-back to the naive result. It also contributes to unnecessarily large regions in the accumulated change mask $\mathbf{M}_A$, and as we will see in Section 6.4, finding a single input image that has a good exposure across a very large region is difficult. In our tests so far, we have found a setting of 25% to work well, providing a good balance between the need to capture all of a moving object, and the need to reduce over-detections which negatively impact the fluid motion fix.

The two parameters determining which regions of the accumulated change mask, $\mathbf{M}_A$, count as fluid motion are t_{MC}, the percentage of times a single pixel must have seen motion to count as a fluid motion pixel, and t_{MFA}, the fraction of a region's pixels in $\mathbf{M}_{AE}$ that must be fluid to count the entire region as fluid. The choice of t_{MFA} should follow similar logic to that of t_{SParea}, and for the results we show it has also been set to 25%. Regarding t_{MC}, with a range of $[0, 1]$, a minimal value will result in all regions in $\mathbf{M}_A$ requiring fluid motion fixes, while a maximum value will result in very few (typically none) of the regions counting as fluid motion. We have already demonstrated the need for a fluid motion fix in the previous chapters, although it may seem that t_{MC} should be set fairly high, e.g., 75%. However, when it is set very high, one type of ghosting scenario can be missed and result in obvious ghosts remaining. Consider a discrete ghost such as a person walking through a scene during capture, but stopping partway through and staying quite still for three or more frames in a row. Deghosting with no fluid motion fix will leave a ghost in this case since during the frames where that ghost stayed still, no change is detected and hence no down-weighting occurs[2]. This scenario is very challenging for change detection-based approaches, and is generally better handled by more advanced global optimization methods, e.g. [33]. However, using an intentionally low t_{MC} will often result in fluid motion detection in these areas, with the result that either the ghost is totally removed, or at least it is fixed in place to reduce blur.

Finally, there are the saturation-related parameters t_{US}, t_{OS}, t_{MFUS}, and t_{MFOS}, determining at what value pixels are considered saturated for normal change detection (t_{US} and t_{OS}) and fluid motion fixing (t_{MFUS} and t_{MFOS}). Section 3.5 explains our choice of $t_{OS} = 250$ as a compromise between the conservative setting used in *pfscalibration* and the maximum value $t_{OS} = 255$ which would result in no pixels counting as over-saturated. For t_{US} and t_{MFUS}, we initially set them symmetrical to t_{OS} and t_{MFOS}, respectively, but

[2]If a discrete ghost stays in place in some region A for just two frames, say frames 3 and 4, it will typically still be caught by change detection without a fluid motion fix since a change is detected in that region between frames 2 and 3, resulting in down-weighting of frame 3, and between 4 and 5, resulting in down-weighting of frame 4.

found that in both cases, reliable data was being thrown away in the dark regions. As such we have lowered them both to 1, which has improved results considerably. This is not surprising when we consider the calibration curve of Figure 3.2, in which the response remains very linear as digital pixel values approach zero, but becomes more erratic as they approach 255. For t_{MFOS}, we argue in Section 5.2 that when a pixel's value in the final HDR composite will be taken from a single input image rather than an average of input images, it should be taken from closer to the center of the response curve to assure that the chosen value is reliable. Early experiments with less restrictive values for t_{MFOS} often resulted in a mild form of the discoloured artifacts shown in Figure 5.6, so we have empirically determined a more restrictive setting of $t_{MFOS} = 0.9Z_{max} \approx 230$ for 8-bit images.

For comparison, the methods of Granados et al. and Zimmer et al. both also require multiple parameters. In the method of Granados et al., the kernel density estimation bandwidth, λ_C, used in the likelihood function, determines over what local range of the input images the likelihood is calculated, and is set experimentally by the authors with reference to a ground truth set. The coefficient γ controls the contribution of the smoothness term, and increasing it reduces the number of labeling regions. The authors show results using $\gamma = 4$ and $\gamma = 12$. The threshold, t_H, for the hard constraint of the smoothness term is set to 5% for their experiments. Two final parameters which must be introduced in the HDR deghosting application, with which we are primarily concerned, are the over- and under-saturation cutoffs above and below which a pixel is assigned an infinite data cost. Zimmer et al., who perform HDR alignment and deghosting using energy-based optical flow, also introduce several parameters. The smoothness term of their energy function is modulated by α, which the authors set to 2 for the results provided. The parameter τ, experimentally set to 0.5, acts as a time step size in the gradient descent scheme used in the energy minimization. A third unnamed parameter is the down-sampling factor for the multi-scale pyramid, which the authors set to 0.95 for their results. This parameter has a large effect on run time; Section 6.2 for details.

6.3.2 Parameter Optimization

Ideally, the parameters introduced here would be rigorously optimized to minimize error of final results with respect to a database of ground truth deghosting results, which could be created either by manual deghosting of images or by simulating (rendering) results. An automatic result would then be compared to the ground truth with some quantitative

measurement, e.g. mean squared error. We leave this for future work because we do not currently have a ground truth set available, and as discussed in Section 4.4, this will prove challenging to create as it is not always clear what consitutes the background.

With ghosts we define as discrete, such as a person who walks rapidly through the scene, it is obvious that such a ghost is not part of the background, so manual labeling could be used to create a ground truth background model. However, even manual labeling is difficult in the case of fluid ghosts. For example, suppose a person stays in one place but gesticulates. That person is clearly not part of the background, so a human performing manual labeling would likely want to remove them. However, if the background is never seen, then the question is what exactly counts as background in this case? Should the gesturing limbs be removed since we see the background behind them at times? If so, we end up with a sliced object, which is a glaring artifact.

With truly fluid objects such as flags or foliage blowing in the wind, it would be unclear even to a human performing manual labeling which input image is most representative of the background, and in reality, any of the input images that is well exposed is a reasonable choice. In such cases, the ground truth would have to have multiple allowable results, and a measure of an automatic method's conformity with the ground truth would have to allow for multiple equally acceptable results. It is also unclear what quantitative measurement would be best in the case of a multi-modal ground truth.

We note, however, that this challenge has not been addressed significantly in the existing literature, and results have typically been assessed purely visually, so this is a good area for future work.

6.4 Negative Example

While the proposed method is successful in many scenarios, some scenarios remain challenging and indicate that there is room for improvement, particularly with respect to the choice of replacement image for the fluid motion fix. One example in which our method has performed poorly is shown in Figure 6.21a. We note that several moving objects are kept, which is not in itself problematic as this can happen with our "best effort" background modeling. However, these objects have been sliced, and now have parts missing. If we consider the proposed fluid motion fix, it is easy to see why this has happened. First, note in Figure 6.21b that the fluid motion region that has been found is very large, indicating that it will be difficult to find a single input image with good exposure of the entire region. The input image of Figure 6.21c, which was chosen

as the best replacement, does indeed have large over-exposed portions, notably on the moving objects themselves. As a result, portions of the moving objects are rejected from the initial replacement image, and are instead drawn from other input images during the iterative search for new best exposures outlined in Section 5.2, with the final result that moving objects are broken up.

If we consider the result of pair-wise down-weighting with no fluid motion fix, shown in Figure 6.21d, we see that in this particular case, it is actually significantly better than the fluid fix result. While this is not generally the case, the result without the fluid fix, calculated before the fluid fix is applied, could be presented to a user as an optional result in case the fluid fix fails, though it would be desirable to avoid user interaction.

Another possible solution comes from noting that while the literature generally suggests choosing replacements for ghosted areas based on exposure [30,31], the LDR version of the fluid fix, introduced in Section 4.4, was successful when comparing to an approximated background model, and Granados et al. use comparisons with a background model, approximated using the naive HDR result, in their data term. As such, we can modify the fluid motion fix to instead compare exposure-normalized input images to the naive HDR result, searching for the lowest mean squared error for the current fluid motion region.

Figures 6.22a, 6.22b, and 6.22c compare the results of the fluid motion fix based on exposure, no fluid motion fix, and fluid motion fix using background comparison, respectively. No fluid motion fix is clearly better than the fix using best exposure in this particular case, and at a macro level, the result with no fluid fix actually appears quite good. On close inspection, as shown in the right hand zoom box, when no fluid motion fix is applied, there are still some subtle ghosting artifacts manifesting here as detached car wheels in the center of the zoom box. In general, ignoring fluid motion areas will leave artifacts much more obvious than these. The result of the fluid motion fix using background comparison improves over that using exposure, and while it does result in the inclusion of some moving objects, they are no longer sliced, and the subtle artifacts resulting from not applying a fluid motion fix are gone.

6.5 Summary

Based on the results shown in this chapter, we find that the proposed method provides similar quality results to those of Granados et al. which, based on our review in Chapter 2, appears to be to the state of the art approach at present. Furthermore, the proposed

method has considerably faster run-times even in a MATLAB implementation, making it more suitable for batch processing of a large number of sets of images as required in the panoramic imaging pipeline outlined in Chapter 1.

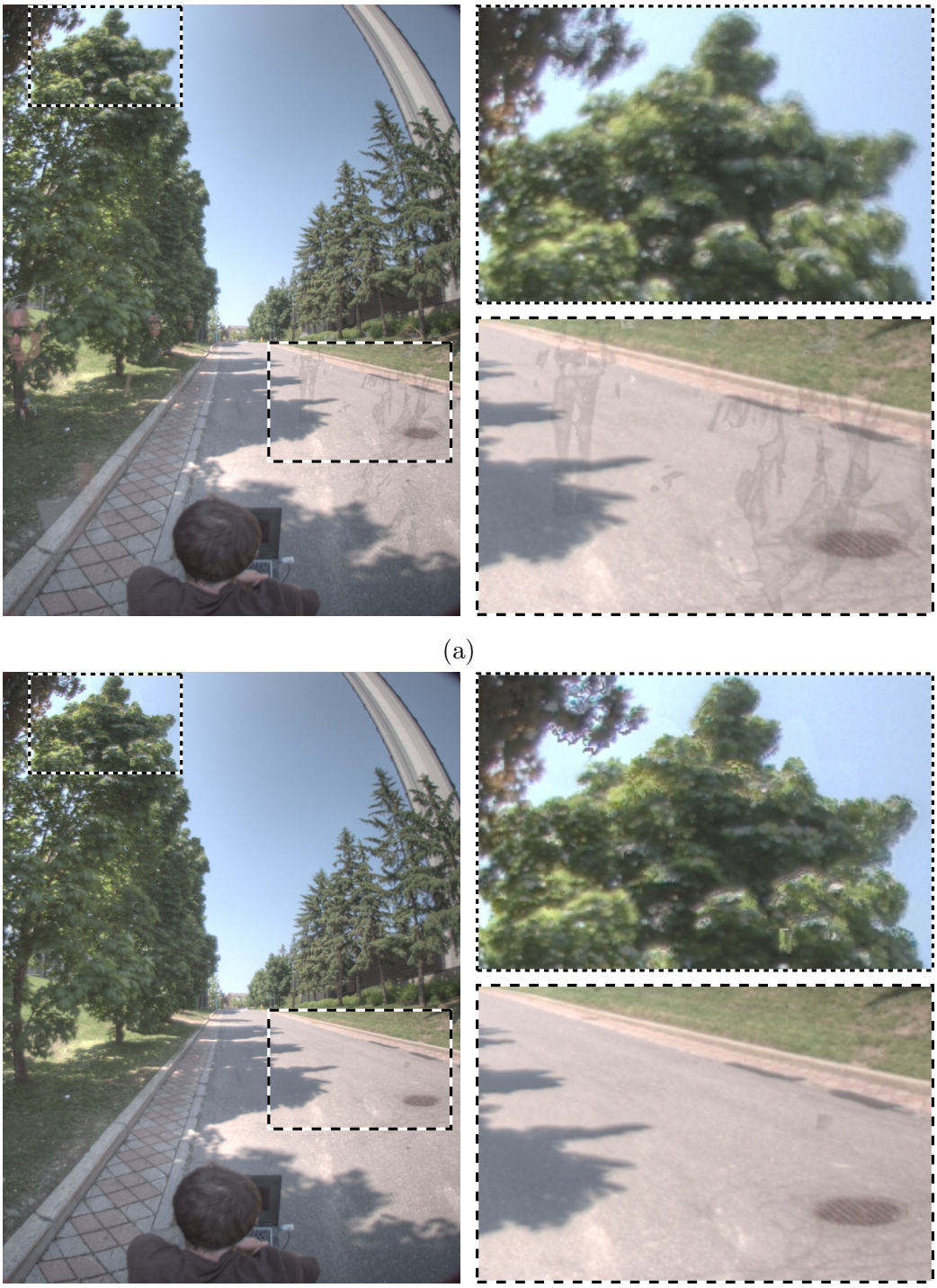


Figure 6.10: Result III. a Naive result. b Result of proposed approach. Discrete ghosts mostly removed. Fluid ghosts sharpened, some small tearing of leaves visible under magnification.

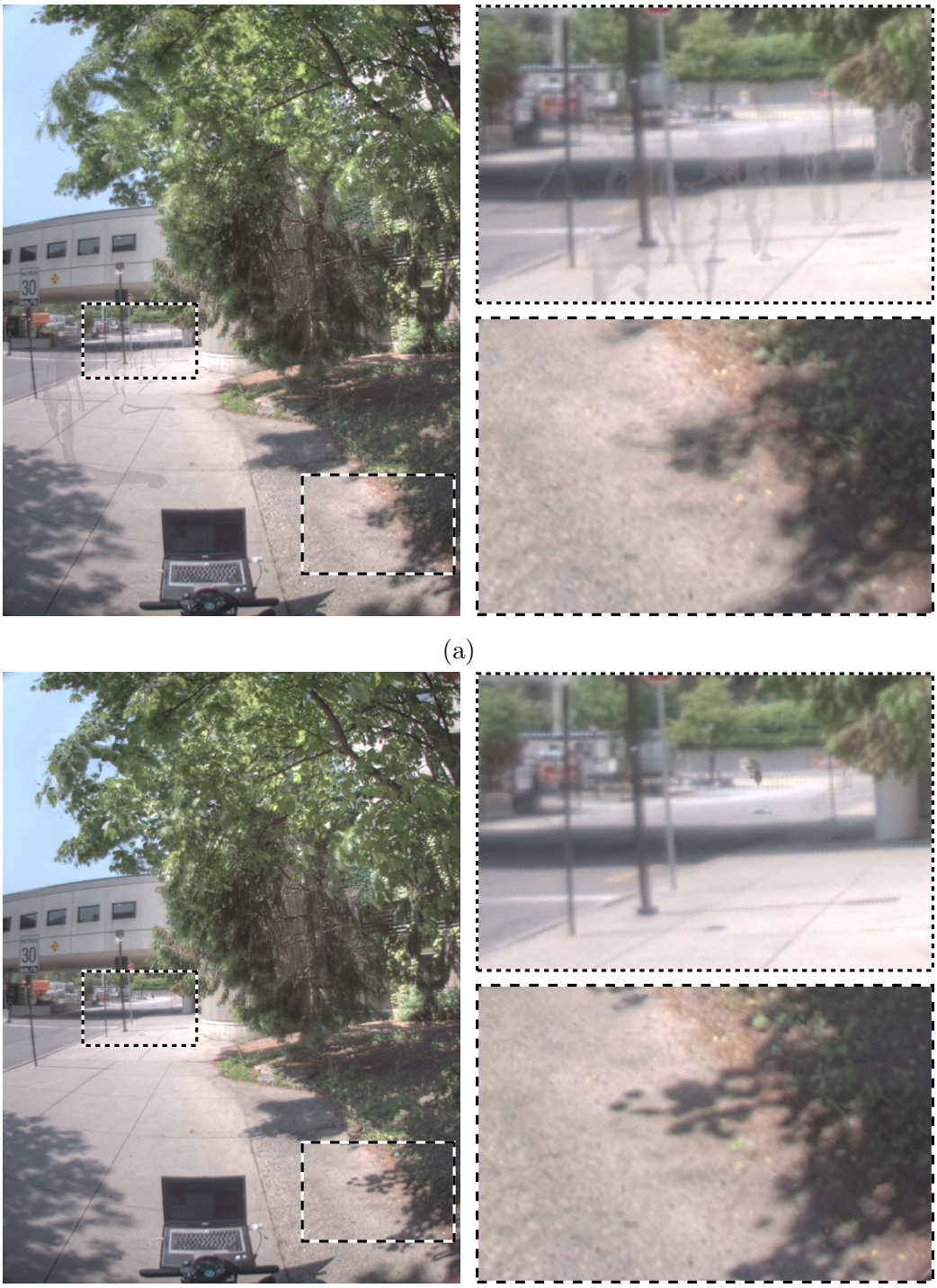
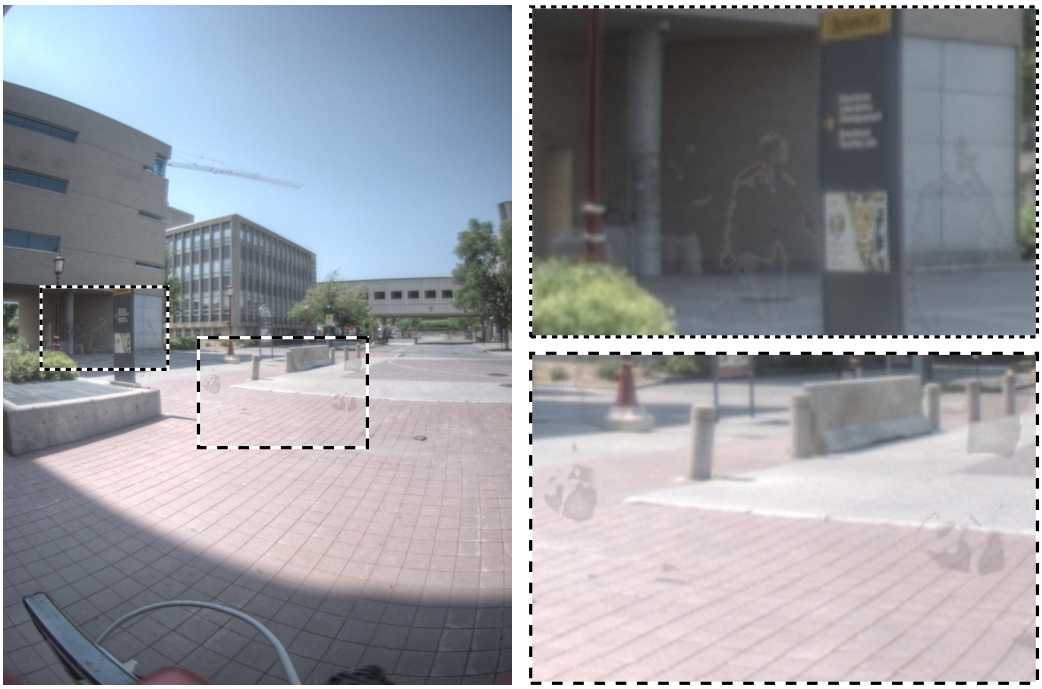
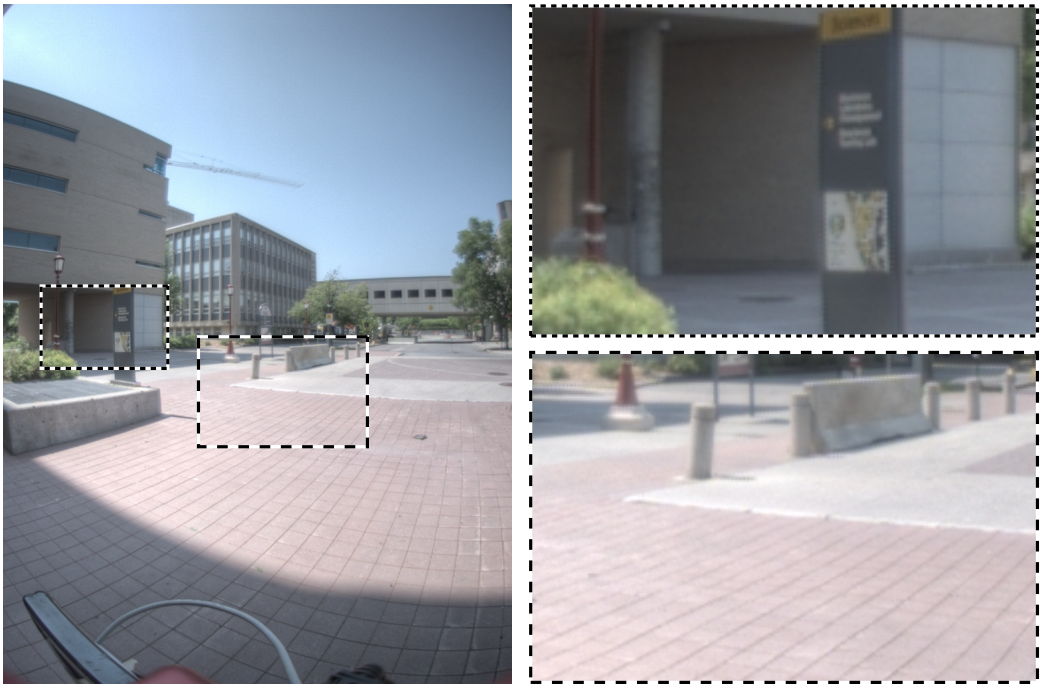


Figure 6.11: Result IV. a Naive result. b Result of proposed approach. Discrete and fluid ghosts both show improvement.



(a)



(b)

Figure 6.12: Result V, showing only discrete ghosting. a Naive result. b Result of proposed approach, with nearly complete ghost removal.

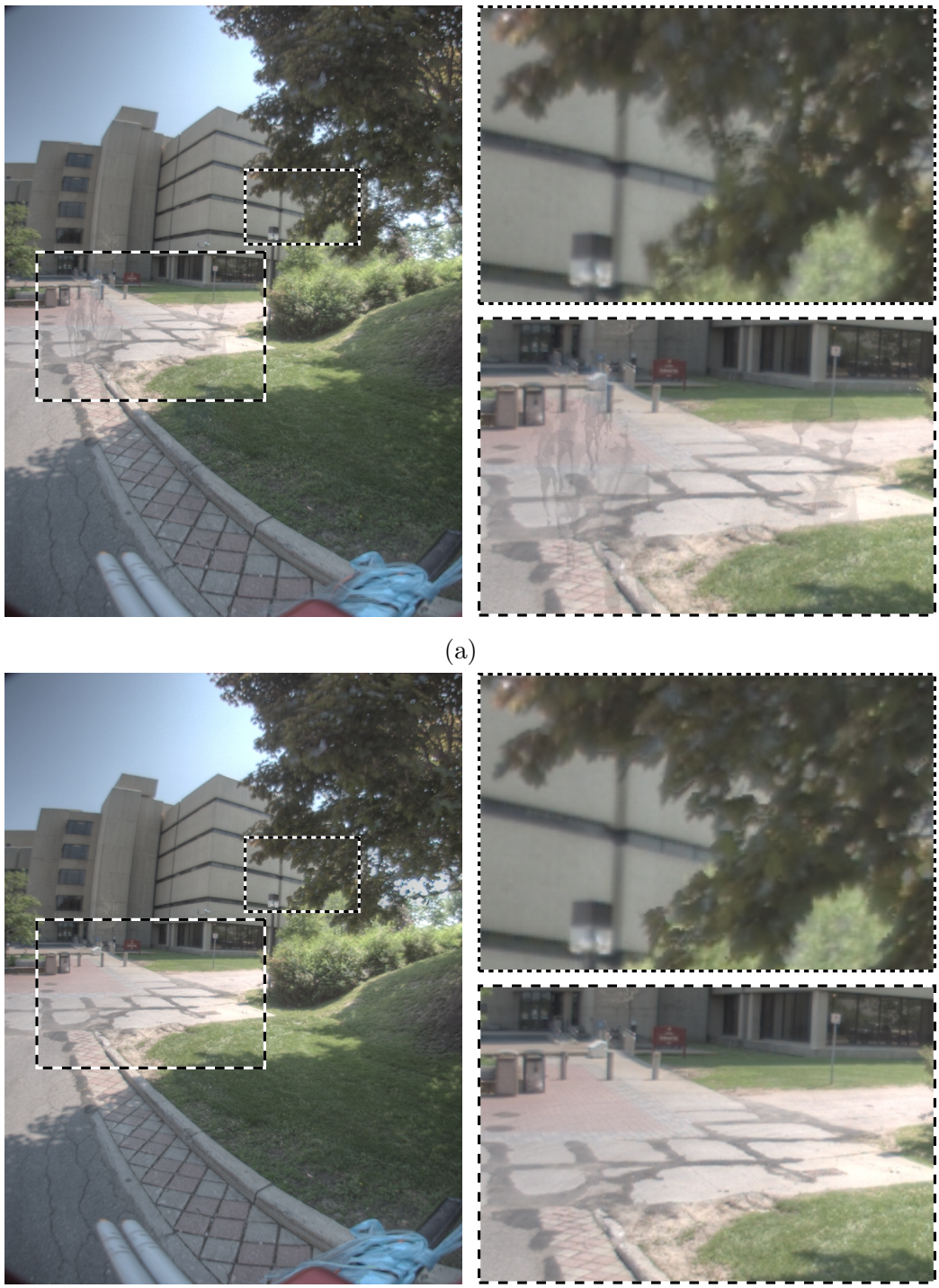
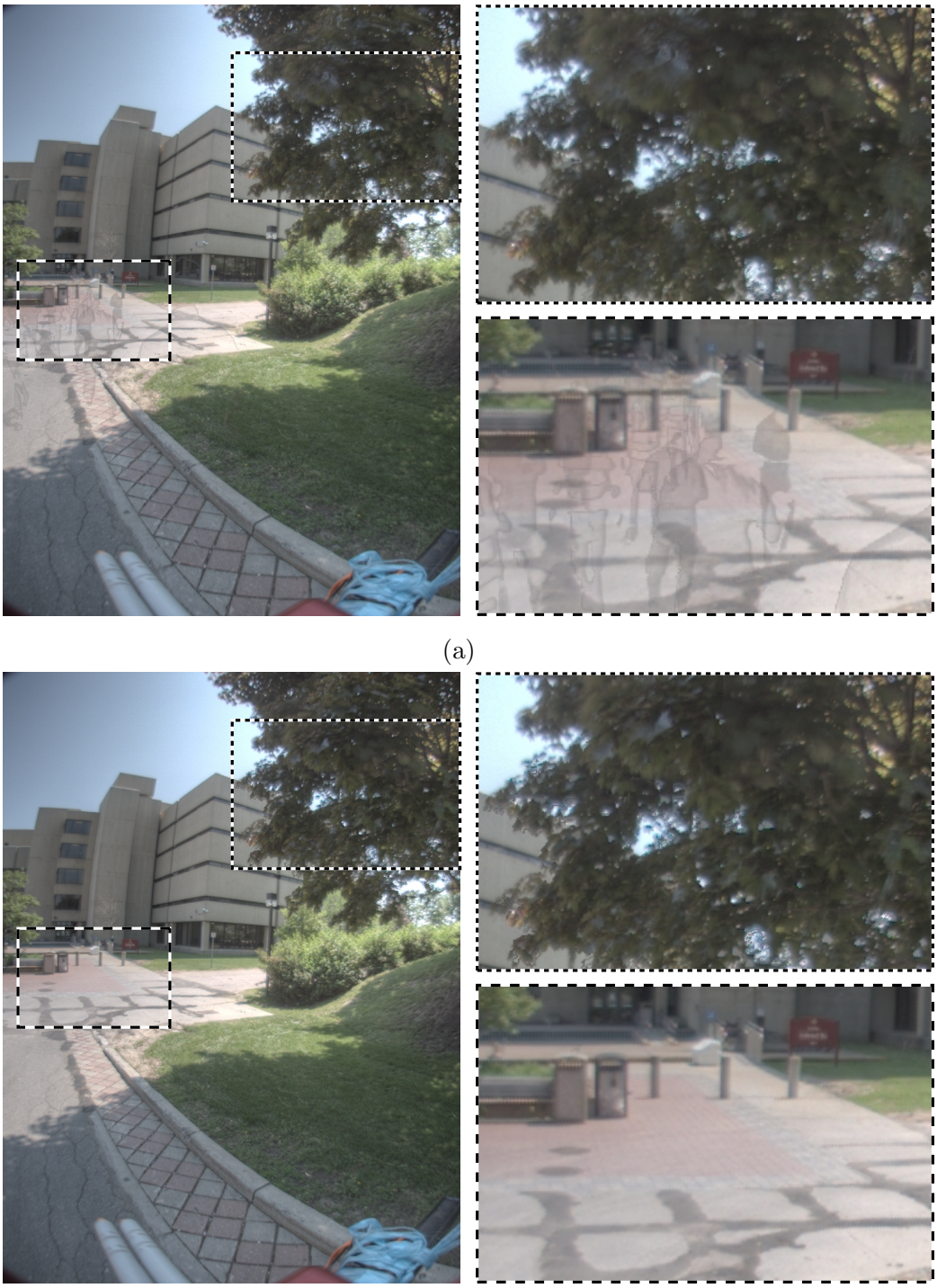


Figure 6.13: Result VI. a Naive result. b Result of proposed approach.

(a)

(b)

Figure 6.14: Result VII. a Naive result. b Result of proposed approach. A tiny ghost remains near the center of the image but occupies less than 0.05% of the image.



(a)

(b)

Figure 6.15: Result VIII. a Naive result. b Result of proposed approach. Some tearing of leaves around the edges of foliage visible under high magnification.

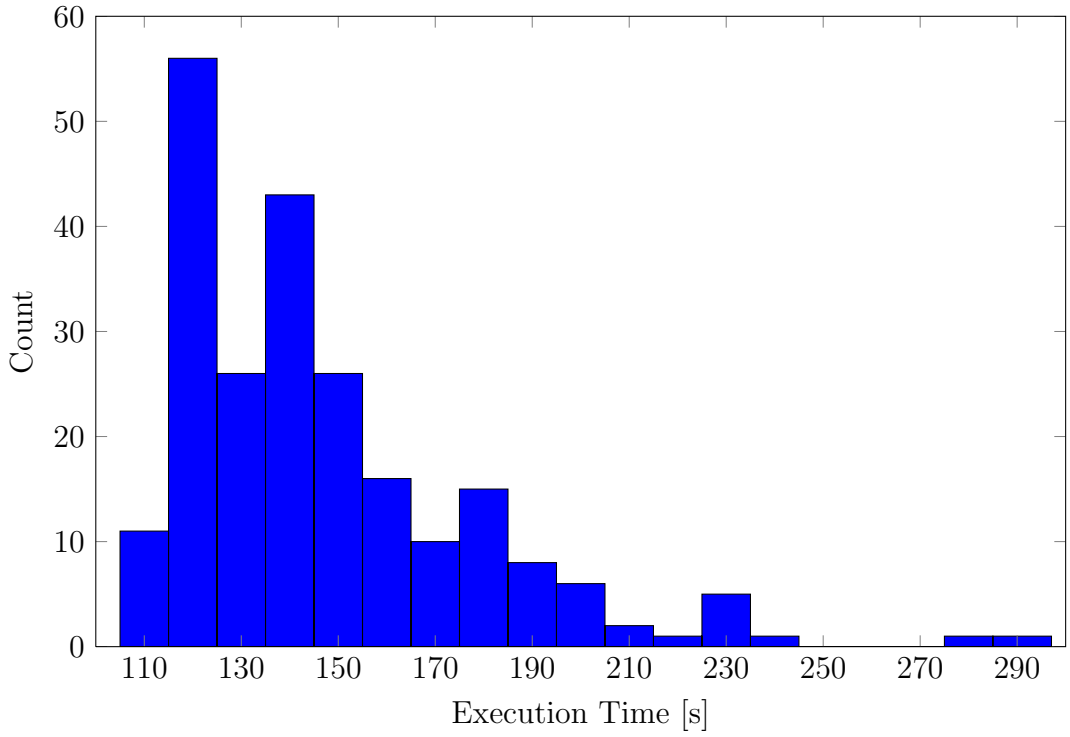
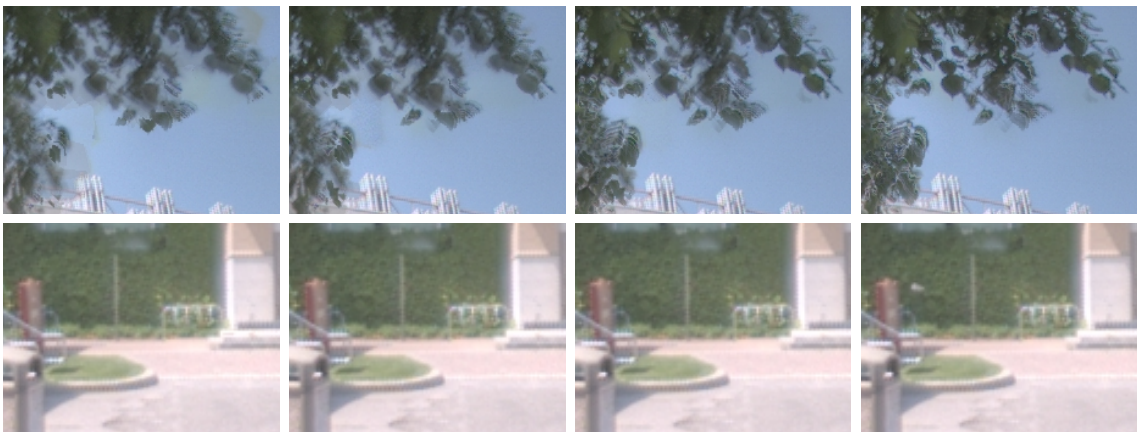
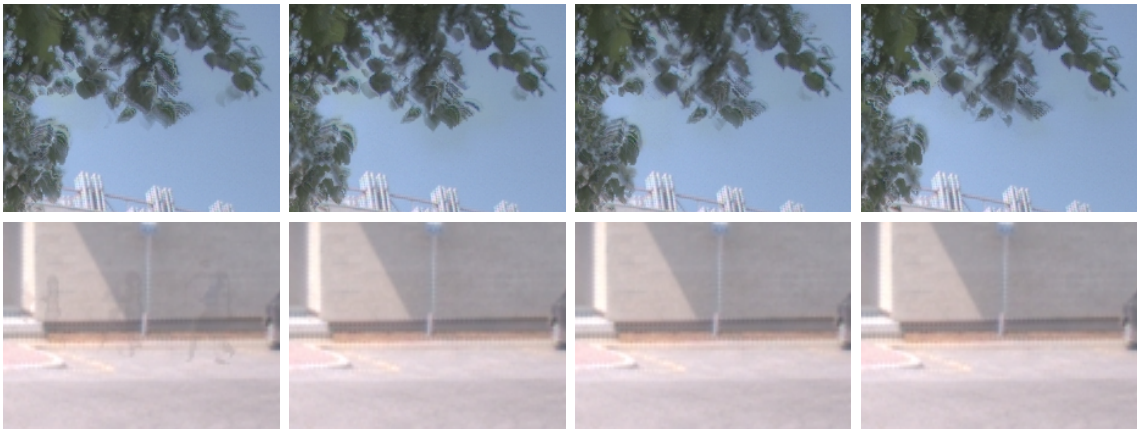


Figure 6.16: A histogram of execution times for deghosting 228 different sets of 16 exposure-bracketed images of resolution 1024x768 using a non-optimized MATLAB implementation of the proposed method. Minimum time is 112s, max is 297s, mean is 147s. 99% of execution times are under 240s, 87% are under 180s, and 64% are under 150s.



$t_C = 0.10$ $t_C = 0.20$ $t_C = 0.40$ $t_C = 0.75$

Figure 6.17: Magnified portions of the Marion Lot data set before fluid motion fix, showing the effects of varying parameter t_C. Leaves start very blurry and with mild gray artifacts like those seen in Figure 5.6. They become more crisp as t_C is increased, but increasing it too much allows portions of ghosts through (bottom row, fourth image, upper-left corner).



$N_{SP} = 400$ $N_{SP} = 800$ $N_{SP} = 1200$ $N_{SP} = 1600$

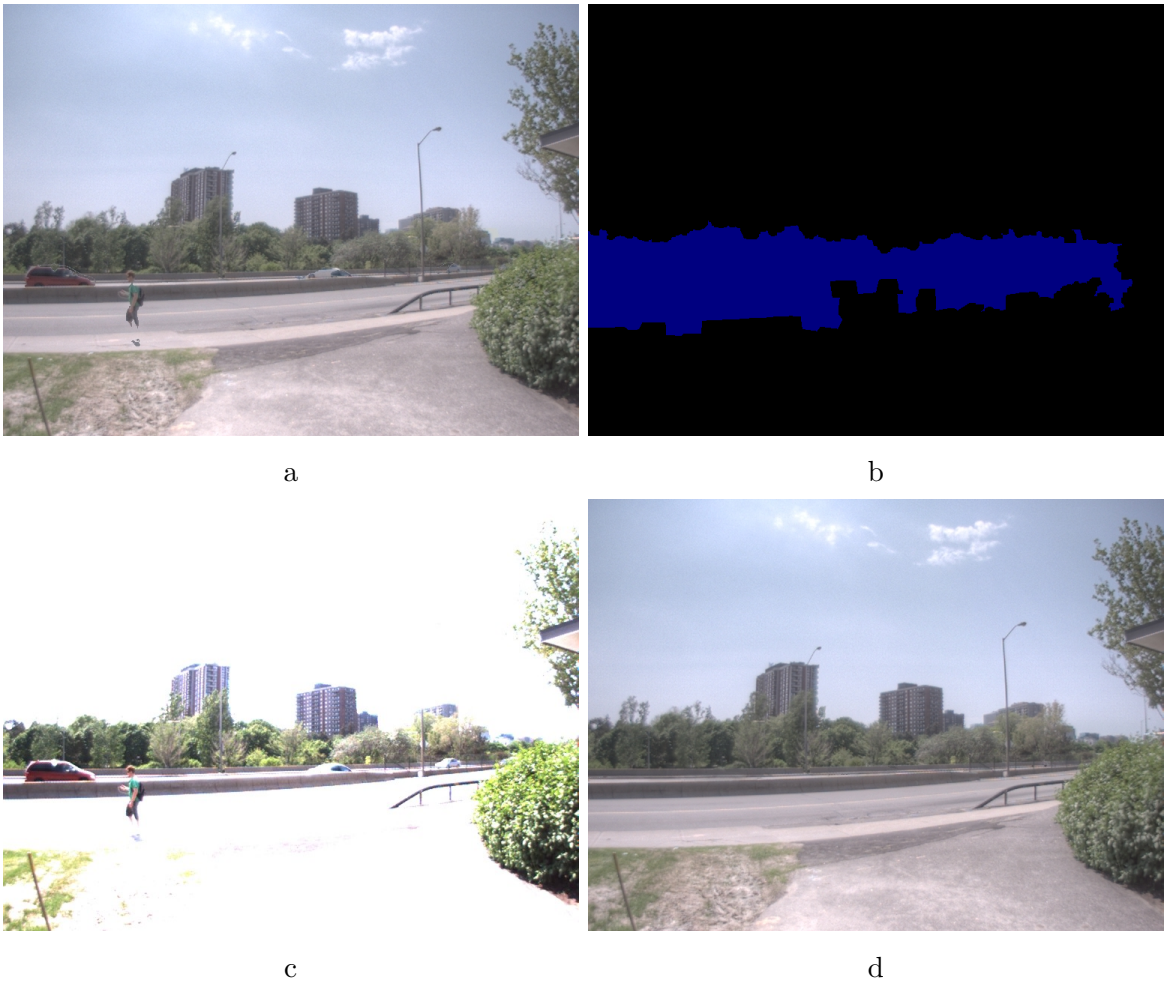
Figure 6.18: Magnified portions of the Marion Lot data set before fluid motion fix, showing the effects of varying parameter N_{SP}. If N_{SP} is set too low, many discrete ghosts remain (bottom row, left). Increasing it makes foliage more crisp (top row). If it is set too high, some large discrete ghosts develop holes (this effect not visible in this data set).

$t_{SParea} = 0.15$ $t_{SParea} = 0.25$ $t_{SParea} = 0.40$ $t_{SParea} = 0.60$

Figure 6.19: Magnified portions of the Marion Lot data set before fluid motion fix, showing the effects of varying parameter t_{SParea}. If t_{SParea} is set too low, foliage is blurry with gray artifacts (top row, left), as with t_C. Increasing it makes foliage more crisp to a point before it becomes blurry again. If it is set too high, portions of some discrete ghosts are kept (bottom row, images 3 and 4).

$t_{MC} = 0.10$ $t_{MC} = 0.25$ $t_{MC} = 0.60$ $t_{MC} = 0.75$

Figure 6.20: Magnified portions of the Marion Lot data set with fluid motion fix, showing the effects of varying parameter t_{MC}. If t_{MC} is set too low, more and larger regions are considered fluid, and large regions are difficult to properly replace in their entirety. Here we see portions of some ghosts remaining (bottom row, left). If it is set too high, fluid regions such as the foliage are not found to be fluid; in this case, the result with $t_{MC} = 0.75$ did not find the foliage to be fluid, so we are left with the basic result (no fluid motion fix), giving blurry leaves.



a b

c d

Figure 6.21: A scenario in which the proposed method struggles. a Result with proposed fluid motion fix, chosing replacement areas for best exposure. b Fluid motion region. c Chosen input image. Note saturated areas, which get rejected, resulting in object slicing. d Result of pair-wise down-weighting without fluid motion fix.

(a)

(b)

(c)

Figure 6.22: Alternate fluid motion fix options. a Pull replacements from best exposed image. b No fluid motion fix. c Pull replacements from best match for background.

Chapter 7

Conclusion

At the outset of this work, our goal was the addition of HDR imaging techniques to a panoramic imaging-based tele-presence system designed for the capture and rendering of various static environments. We have outlined a custom capture program for capturing exposure-bracketed images on the Ladybug2 panoramic camera, a new fast deghosting approach enhanced by cutting edge over-segmentation techniques and a novel fix for fluid motion areas. We have shown the success of our system, which improves image quality over the LDR version, and our deghosting method, which produces better results than commercial software, and similar results to state-of-the-art methods, with much faster run-times. Here we provide a summary of our work, discuss limitations of the proposed method and HDR capture in general, and propose future work to further enhance our system.

7.1 Summary

The goal in adding HDRI to the tele-presence system was to enhance the system's ability to render environments with high visual fidelity, thereby enhancing the user's sense of "being there." To this end, we created a new custom exposure-bracketed image capture program for the Ladybug2 panoramic camera. Since capture will frequently happen while objects are moving in the scene, we devoted significant effort to our proposed deghosting method. Another important goal was high speed processing because the tele-presence system is used to capture data sets of hundreds or even thousands of images.

We started by reviewing the existing relevant literature, including classic HDR literature for performing naive HDR combination, various categories of deghosting methods,

and a brief review of segmentation and tonemapping. The third chapter outlined our custom capture utility, including a manual mode for advanced uses requiring control over the number of images captured, and an automated method requiring less user input.

Next, we introduced our proposed deghosting method. With the goal of fast processing of large data sets in mind, we revisited change detection-based ghost removal, chosen for its low complexity. We used a reasoned consideration of the information available when exposure is varied in a controlled manner, and demonstrated the use of recent superpixel segmentation methods for enhancing the results of change detection. The resulting inter-frame change masks were used to reduce the contribution of changed areas to the HDR weighted average on a per-frame basis, removing discrete ghosts while maintaining high dynamic range information available in the rest of the input image stack. We also introduced a novel fluid motion fix for areas that see motion throughout a large portion of the input images. These areas are replaced with information from a single judiciously chosen input image so that fluid ghosts like foliage blowing in the wind or groups of people milling about on spot are "frozen" to remove blurring. We outlined our approach first using an analogous low dynamic range version, followed by a separate outline of the additional steps required for its application to HDR deghosting.

Finally, we demonstrated the competitive performance of our approach, which even in a non-optimized MATLAB implementation produces markedly better results than leading commercial HDR software. The proposed method also produces results comparable to state of the art graph cut and optical flow approaches, with significantly faster processing times, as detailed in our presented results.

Our key contributions to the literature are:

- Change detection in exposure-bracketed sequences based on a deliberate logical analysis of how pixel values should behave within an exposure-bracketed image stack and what can be known about pixels with respect to under-/over-exposure, and known exposure times.

- Use of recent Superpixel segmentation techniques to refine change detection results.

- Classification of ghosts as either "discrete" (ghosts that displace by large amounts between each frame) or "fluid" (ghosts that stay in one place but visually deform or fluctuate, such as foliage or flags blowing in the wind).

- A fast novel technique for detecting fluid areas and choosing a single input image to replace them so that ghosts in those regions are frozen.

7.2 Limitations

Some limitations of the proposed deghosting method can be seen. In comparison to the work of Granados et al. [33], we find that on a whole-image level we produce results of similar or better quality in less time. However, their method is able to correct certain types of very subtle fluid ghosting that our method does not pick up on. This is particularly noticeable in foliage that has only experienced a mild breeze: while our method does not see any change, Granados et al. are able to correct this mild motion and produce more crisp foliage in areas like this.

Our method also only provides a "best effort" background model with short processing time. As such, in some challenging scenarios, notably when the background is highly obscurred due to discrete ghosts and only seen once during capture, the method of Granados et al. could theoretically still model the background. In these scenarios our method will generally recognize fluid motion and attempt a fix. However, it is less likely than Granados et al. to choose the one image that saw the background, though it will still at least freeze motion in such areas, a good compromise when time is constrained.

Regarding data set size, existing methods for ghost removal typically look at image sets of 10 or more images. In our experiments we have tested image sets of 10-25 images. However, as described in Section 2.1.3, there is a growing application for HDR images created from just 2-3 images due to the inclusion of integrated HDR functionality on mobile platforms such as cellphones and digital cameras. Such small image sets make it difficult to build statistical relationships between the images, indicating that there is a lower bound of image set size for most existing methods, as well as our own. In such cases, the state of the art methods [29, 33] may be more applicable despite their increased computational cost. Notably, with just 2-3 images captured in rapid succession, the limitations and computational cost of the optical flow method of Zimmer et al. [29] (see Section 2.3.1) would likely be less prominent, while the resulting small displacements of objects may allow it to successfully realign even discrete ghosts.

In addition, several hardware limitations can hamper the ability of an HDR pipeline to extend dynamic range. Some of these cannot be avoided, while some may be candidates for further work:

- **Camera Shutter Speeds:** For a camera with a fixed gain and some minimum exposure exposure time Δt_{min}, if an image captured at Δt_{min} contains no under-saturated pixels (i.e. all pixels are above zero), then we have already captured as much of the dynamic range of that particular scene as we can with that camera. To

see why this is true, consider that the dynamic range of a single LDR image with linear response is the ratio of the brightest to the darkest points in the image. As we increase shutter speeds from Δt_{min}, the values will remain linearly related and will simply scale with Δt until the brightest values saturate, at which point the ratio between max and min values will begin to decrease. We have found this to be a limitation with the Ladybug2 in some brightly sunlit outdoors scenes, where we occasionally find that the darkest image has no under-saturated pixels. In such scenarios, the proposed method still has the advantages of modeling the background and reducing noise. A possible fix for this issue would be to add neutral density filters (shaded filters) to the camera, although this task is not a trivial since the camera uses micro-lenses that lack standard threading for screw-on filters.

- **CCD Smear:** CCD smear can occur when a CCD-based camera with no mechanical shutter is exposed to an extremely bright light source such as direct sunlight. CCDs only have readout electronics at the end of every scanline, so pixel values for a scanline need to be read one at a time, and they are shifted or "walked" off the scanline as each pixel is read. CCDs without mechanical shutters generally use an interline frame transfer configuration in which a scanline of pixels is exposed to the scene, then shifted sideways onto a "vertical transfer register," which is a secondary scanline covered with a light-resistant mask. This avoids the smearing artifacts that result if the pixels remain exposed to light as they are shifted towards the reader. However, extremely bright light sources can still penetrate or bleed under the mask. When this occurs, the entire transfer register becomes saturated both above and below the light source, with the result being a pronounced white line. An example of CCD smear is shown in Figure 7.1a and more information is available in [113].

- **Veiling Glare:** Very bright light sources, which may or may not be in the field of view of the camera, can cause stray light to scatter within the camera optics and sometime give rise to a phenomenon known as veiling glare. In the context of HDR imaging, this sometimes creates scenarios where a bright light source next to a dark region makes it impossible to capture the dark region well because, as exposure is increased, the veiling glare washes out the dark region. A example of veiling glare is shown in Figure 7.1b.

a b

Figure 7.1: Limitations of the hardware can sometimes hamper capture. a CCD smear. b Veiling glare (visible in upper left quarter of image). Both images are tonemapped HDR results from our experiments.

7.3 Future Work

The following are several directions of interest for improvements or enhancement to our HDR tele-presence system:

- **Camera Motion and Image Alignment:** Currently, we assume a static camera for our HDR composition, so the capture involves stopping at each point where we want to capture an HDR image. This can be tedious when the capture rig is operated by a human user. A major enhancement to the proposed system would be the integration of an alignment technique, allowing image capture from a moving platform. Various methods have been proposed [28,29,46], and notably, the optical flow method of Zimmer et al. [29] provides powerful alignment capabilities. Their result still requires a good deghosting method as a post-processing stage since they do not handle large discrete ghosts, so it would be a natural fit for our system.

- **Optimal Capture:** Our capture of exposure-bracketed images relies on the popular knowledge that bracketing is best accomplished by fixing gain (ISO) while

varying exposure time. Recent work by Hasinoff et al. [114] on noise-optimal/ time-optimal capture has shown that more optimal capture can be achieved by considering a detailed model of noise sources in digital cameras, by varying the ISO in addition to exposure time, and by using high ISO settings for some of the images. They show that as few as three images using simultaneously varied exposure time and ISO can produce the same dynamic range and signal to noise ratio as much larger sets using traditional exposure-bracketing. This would be beneficial from a data storage point of view, and also an image alignment perspective, since fewer images would need to be aligned. Similar work by Granados et al. [115] has investigated optimal weighting functions based on a photometric camera calibration model that factors in noise sources. The resulting model can also be used to choose optimal exposure settings and to denoise HDR images.

- **CCD Smear:** CCD Smear was regularly encountered in our experiments, particularly on the top CCD of the Ladybug2 when capturing outdoors since it is often exposed to sunlight. Furthermore, when a set of exposure-bracketed images experiences CCD smear, it occurs in all the images and gets worse as exposure increases, so it is not an option to simply look for an image which did not experience it. A naive HDR approach helps somewhat by down-weighting the contributions of brighter images in which the smear line is wider. However, the resulting HDR image will still contain the smear from the least affected image, as shown in the tonemapped HDR result of Figure 7.1b. A temporary solution to this problem is to avoid capturing with any of the CCDs exposed to direct sunlight. A possible long-term solution is to develop a method to detect and inpaint areas affected by CCD smear. Detection would be aided by the facts that smear is always fully saturated in all three channels and therefore bright white, and CCD smear is always parallel to scanlines. It therefore has a predictable colour and shape, though the width may vary. If the area affected by CCD smear can be detected, it is then a good candidate for inpainting since in outdoor scenes it often occurs in areas of the sky with simple patterns or no pattern at all, and it is also usually a thin line rather than a large area.

- **Parameter Optimization:** Our proposed method involves multiple parameters. While methods with no or few parameters are attractive due to their simplicity of use for end-users, when they fail, there is little room for adjustment. At the same time, highly parameterized methods can be confusing for end users. As such,

we would like to perform an optimization of our parameters for several intuitively-named broad classes of ghosting, e.g. "people and cars", "foliage", etc., so that these could be presented to an end user. In an advanced interface, the parameters themselves could also be presented for tweaking, again with intuitive names. For example, the number of superpixels could be adjusted via a slider simply indicating "large ghosts" on one end and "small ghosts" on another, corresponding to low and high values for N_{SP}. Another possible avenue would be to integrate a high-level scene descriptor such as Gist [116] to classify regions of the image into categories semantically meaninful in a deghosting context, like those mentioned above, so that different images and/or regions could be automatically processed with parameter settings that have been found to work best for those particular types of ghosting. Such a descriptor might also be used to choose between the different fluid motion fix options discussed in Section 6.4.

Bibliography

[1] (2002, Jul.) Gustave Le Gray, photographer. The J. Paul Getty Museum. [Online]. Available: http://www.getty.edu/art/exhibitions/le_gray/

[2] P.E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *SIGGRAPH*, 1997, pp. 369–378.

[3] "OpenEXR," Online, Industrial Light and Magic, Jul. 2010. [Online]. Available: http://www.openexr.com/

[4] (2010, Feb.) Product catalog - spherical. Point Grey Research Incorporated. Vancouver, British Columbia. [Online]. Available: http://www.ptgrey.com/products/Point_Grey_spherical_catalog.pdf

[5] A.J.P. Theuwissen, *Solid-State Imaging with Charge-Coupled Devices*. Springer, Mar. 1995, vol. 1.

[6] T. Mitsunaga and S.K. Nayar, "Radiometric self calibration," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1. Los Alamitos, CA, USA: IEEE Computer Society, 1999, p. 1374.

[7] M.A. Robertson, S. Borman, and R.L. Stevenson, "Dynamic range improvement through multiple exposures," in *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, vol. 3. IEEE, 1999, pp. 159–163.

[8] S.K. Nayar and T. Mitsunaga, "High dynamic range imaging: Spatially varying pixel exposures," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1. IEEE, 2000, pp. 472–479.

[9] S. Nayar and V. Branzoi, "Adaptive dynamic range imaging: Optical control of pixel exposures over space and time," in *Proceedings of the Ninth IEEE Interna-*

tional Conference on Computer Vision-Volume 2. IEEE Computer Society, 2003, p. 1168.

[10] M. Aggarwal and N. Ahuja, "Split aperture imaging for high dynamic range," *International Journal of Computer Vision*, vol. 58, no. 1, pp. 7–17, 2004.

[11] E. Ikeda, "Image data processing apparatus for processing combined image signals in order to extend dynamic range," US Patent 5,801,773, Sep., 1998.

[12] R. Street, "High dynamic range segmented pixel sensor array," US Patent 5,789,737, Aug., 1998.

[13] V. Brajovic, R. Miyagawa, and T. Kanade, "Temporal photoreception for adaptive dynamic range image sensing and encoding," *Neural Networks*, vol. 11, no. 7-8, pp. 1149–1158, 1998.

[14] J. Burghartz, H. Graf, C. Harendt, W. Klingler, H. Richter, and M. Strobel, "HDR CMOS imagers and their applications," in *Solid-State and Integrated Circuit Technology, 8th International Conference on*. IEEE, 2006, pp. 528–531.

[15] A. El Gamal. (2002) High dynamic range image sensors. Tutorial presented at 2002 IEEE International Solid-State Circuits Conference. [Online]. Available: http://www-isl.stanford.edu/~abbas/group/papers_and_pub/isscc02_tutorial.pdf

[16] T. Yamada, S. Kasuga, T. Murata, and Y. Kato, "A 140db-dynamic-range MOS image sensor with in-pixel multiple-exposure synthesis," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*. IEEE, Feb. 2008, pp. 50–594.

[17] J. Solhusvik, S. Yaghmai, A. Kimmels, C. Stephansen, A. Storm, J. Olsson, A. Rosnes, T. Martinussen, T. Willassen, P.O. Pahr, S. Eikedal, S. Shaw, R. Bhamra, S. Velichko, D. Pates, S. Datar, S. Smith, L. Jiang, D. Wing, and A. Chilumula, "A 1280x960 3.75um pixel CMOS imager with triple exposure HDR," in *International Image Sensor Workshop*. Bergen, Norway: Aptina Imaging, 2009.

[18] B. Fowler, "High dynamic range image sensor architectures," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 7876, 2011, p. 1.

[19] S. Kavusi and A. El Gamal, "Quantitative study of high dynamic range image sensor architectures," in *SPIE proceedings series*. Society of Photo-Optical Instrumentation Engineers, 2004, pp. 264–275.

[20] J. van Rooy, P. Centen, and M. Stekelenburg. (2002, Sep.) Viper filmstream camera: A technical overview. Online Documentation Archive. Grass Valley USA. Nevada City. [Online]. Available: http://www.grassvalley.com/docs/Miscellaneous/cameras/viper/technical_overview.pdf

[21] (2008, Oct.) New avocet CMOS image sensor technology and products from sensata technologies for automotive and vehicle safety systems. Press Release. Sensata Technologies. [Online]. Available: http://www.sensata.com/about/rel96.htm

[22] (2011, May) Spherocam HDR. Spherocam VR. [Online]. Available: http://www.spheron.com/en/intruvision/solutions/spherocam-hdr.html

[23] D. Bradley, A. Brunton, M. Fiala, and G. Roth, "Image-based navigation in real environments using panoramas," in *IEEE International Workshop on Haptic Audio Environments and their Applications*, 2005.

[24] M. Beermann and E. Dubois, "Acquisition processing chain for dynamic panoramic image sequences," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 5, 16 2007-oct. 19 2007, pp. V –217 –V –220.

[25] D. Wojtaszek and J. Saboune, "Capturing panoramic images using the scooter and ladybug camera," Jun. 2009, internal document on image capture using the Ladybug2 camera and mobility scooter.

[26] S. Mann and R.W. Picard, "On being 'undigital' with digital cameras: Extending dynamic range by combining differently exposed pictures," in *Proceedings of IS&T*, 1995, pp. 442–448.

[27] L. Bogoni, "Extending dynamic range of monochrome and color images through fusion," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 3. IEEE, 2000, pp. 7–12.

[28] S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High dynamic range video," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 319–325, 2003.

[29] H. Zimmer, A. Bruhn, and J. Weickert, "Freehand HDR imaging of moving scenes with simultaneous resolution enhancement," in *Computer Graphics Forum*, vol. 30, no. 2. Wiley Online Library, 2011, pp. 405–414.

[30] E. Reinhard, G. Ward, S. Pattanaik, and P.E. Debevec, *High Dynamic Range Imaging*, 1st ed. San Francisco, CA: Morgan Kaufmann, 2005.

[31] K. Jacobs, C. Loscos, and G. Ward, "Automatic high dynamic range image generation for dynamic scenes," *IEEE Computer Graphics and Applications*, vol. 28, no. 2, pp. 84–93, 2008.

[32] O. Gallo, N. Gelfand, W.C. Chen, M. Tico, and K. Pulli, "Artifact-free high dynamic range imaging," in *Computational Photography (ICCP), IEEE International Conference on*. IEEE, 2009, pp. 1–7.

[33] M. Granados, H.P. Seidel, and H.P.A. Lensch, "Background estimation from non-time sequence images," in *Proceedings of Graphics Interface 2008*. Canadian Information Processing Society, 2008, pp. 33–40.

[34] X. Chen, Y. Shen, and Y.H. Yang, "Background estimation using graph cuts and inpainting," in *Proceedings of Graphics Interface 2010 on Proceedings of Graphics Interface 2010*. Canadian Information Processing Society, 2010, pp. 97–103.

[35] T. Ratcliff, *A World in HDR*, 1st ed., R. Gulick, Ed. New Riders, 2010.

[36] (2011, May) Spherocam. Spherocam VR. [Online]. Available: http://www.spheron.com/en/intruvision/solutions/spherocam-hdr/spherocam.html

[37] T. Mertens, J. Kautz, and F. Van Reeth, "Exposure fusion," in *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*. IEEE, 2007, pp. 382–390.

[38] B. Madden, "Extended intensity range imaging," *Technical Reports (CIS)*, p. 248, 1993.

[39] (2010) HDR images in photography - about dynamic range, tone mapping and HDR imaging for photography. HDRsoft. [Online]. Available: http://www.hdrsoft.com/resources/dri.html#calib

[40] E.A. Khan, A.O. Akyüz, and E. Reinhard, "Ghost removal in high dynamic range images," in *Image Processing, IEEE International Conference on*. IEEE, 2006, pp. 2005–2008.

[41] J. Choi, M.K. Park, and M.G. Kang, "High dynamic range image reconstruction with spatial resolution enhancement," *The Computer Journal*, vol. 52, no. 1, p. 114, 2009.

[42] B.K. Gunturk and M. Gevrekci, "High-resolution image reconstruction from multiple differently exposed images," *Signal Processing Letters, IEEE*, vol. 13, no. 4, pp. 197–200, 2006.

[43] H. Zimmer, A. Bruhn, and J. Weickert. (2010, Dec.) Freehand HDR imaging of moving scenes with simultaneous resolution enhancement. Mathematical Image Analysis Group, Saarland University. Germany. [Online]. Available: http://www.mia.uni-saarland.de/Research/SR-HDR/index.shtml

[44] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, pp. 285–296, 1975.

[45] R. Lukac, Ed., *Computational Photography: Methods and Applications*, 1st ed., ser. Digital Imaging and Computer Vision Series. CRC Press, 2011.

[46] G. Ward, "Fast, robust image registration for compositing high dynamic range photographs from handheld exposures," *Journal of Graphics Tools*, vol. 8, pp. 17–30, 2003. [Online]. Available: http://www.anyhere.com/gward/papers/jgtpap2.pdf

[47] M. Piccardi, "Background subtraction techniques: a review," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 4, Oct. 2004, pp. 3099 – 3104 vol.4.

[48] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *iccv*. Published by the IEEE Computer Society, 1999, p. 255.

[49] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 780–785, 1997.

[50] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*, 1st ed. O'Reilly Media, 2008.

[51] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2. IEEE, 1999.

[52] P. Power and J. Schoonees, "Understanding background mixture models for foreground segmentation," in *Proceedings Image and Vision Computing New Zealand*, vol. 2002. Citeseer, 2002.

[53] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," *Computer VisionECCV 2000*, pp. 751–767, 2000.

[54] B. Lo and S. Velastin, "Automatic congestion detection system for underground platforms," in *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on.* IEEE, 2001, pp. 158–161.

[55] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-time imaging*, vol. 11, no. 3, pp. 172–185, 2005.

[56] E. Reinhard, E. Khan, A.O. Akyüz, and G. Johnson, *Color imaging: fundamentals and applications*, 1st ed. AK Peters Ltd, 2008.

[57] D.L. Ruderman, T.W. Cronin, and C.C. Chiao, "Statistics of cone responses to natural images: implications for visual coding," *Journal of the Optical Society of America A: Optics, Image Science, and Vision*, vol. 15, pp. 2036–2045, Aug. 1998.

[58] S. Cohen, "Background estimation as a labeling problem," in *Tenth IEEE International Conference on Computer Vision*, vol. 2. IEEE Computer Society, 2005, pp. 1034–1041.

[59] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen, "Interactive digital photomontage," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 294–302, 2004.

[60] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 9, pp. 1124–1137, 2004.

[61] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1068–1080, 2008.

[62] R. Gonzalez, R. Woods, and S. Eddins, *Digital Image Processing Using MATLAB*, 1st ed. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 2003.

[63] S. Eddins. (2002, Feb.) The watershed transform - strategies for image segmentation. The MathWorks. Natick, MA. [Online]. Available: http://www.mathworks.com/company/newsletters/news_notes/win02/watershed.html

[64] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[65] S. Beucher and C. Lantuejoul, "Use of watersheds in contour detection. int," in *Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation*, vol. 17, no. 21. Rennes, France: Centre de Géostatistique et de Morphologie Mathématique, Sep. 1979, pp. 2–1.

[66] F. Meyer, "Topographic distance and watershed lines," *Signal Processing*, vol. 38, pp. 113–125, Jul. 1994. [Online]. Available: http://dx.doi.org/10.1016/0165-1684(94)90060-4

[67] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, 2nd ed. Upper Saddle River, New Jersey: Prentice Hall, 2002.

[68] (2010) Marker-controlled watershed segmentation. The MathWorks. Natick, MA. [Online]. Available: http://www.mathworks.com/products/demos/image/watershed/ipexwatershed.html

[69] S. Eddins. (2006, Jun.) Cell segmentation. The Mathworks Inc. [Online]. Available: http://blogs.mathworks.com/steve/2006/06/02/cell-segmentation/

[70] ——. (2009, May) Locating the US continental divide, part 2 - watershed transform. The Mathworks Inc. [Online]. Available: http://blogs.mathworks.com/steve/2009/05/01/continental-divide-2-watershed/

[71] P. Soille, *Morphological Image Analysis: Principles and Applications*, 2nd ed. Secaucus, NJ: Springer-Verlag, 2003, corrected Second Printing.

[72] X. Ren and J. Malik, "Learning a classification model for segmentation," in *In Proceedings Ninth IEEE International Conference on Computer Vision*, vol. 1. Nice: IEEE, 2003, pp. 10–17.

[73] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *IEEE Conf. Computer Vision and Pattern Recognition*, 1997, pp. 731–7.

[74] J. Malik, S. Belongie, T. Leung, and J. Shi, "Contour and texture analysis for image segmentation," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 7–27, 2001.

[75] G. Mori, "Guiding model search using segmentation," in *Computer Vision, Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1417–1423.

[76] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi, "Turbopixels: Fast superpixels esing geometric flows," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 12, pp. 2290–2297, 2009.

[77] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC Superpixels," École Polytechnique Fédéral de Lausssanne (EPFL), Tech. Rep. 149300, Jun. 2010.

[78] K. Devlin, "A review of tone reproduction techniques," Department of Computer Science, University of Bristol, Tech. Rep. CSTR-02-005, 2002.

[79] F. Drago, K. Myszkowski, T. Annen, and N. Chiba, "Adaptive logarithmic mapping for displaying high contrast scenes," in *Computer Graphics Forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 419–426.

[80] C. Schlick, "Quantization techniques for visualization of high dynamic range pictures," in 5^{th} *Eurographics Workshop on Rendering*. Springer-Verlag, 1994, pp. 7–20.

[81] J. Tumblin and H. Rushmeier, "Tone reproduction for realistic images," *Computer Graphics and Applications, IEEE*, vol. 13, no. 6, pp. 42 –48, Nov. 1993.

[82] G. Larson, H. Rushmeier, and C. Piatko, "A visibility matching tone reproduction operator for high dynamic range scenes," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 3, no. 4, pp. 291–306, 1997.

[83] G. Ward, "A contrast-based scalefactor for luminance display," in *Graphics gems IV*. Academic Press Professional, Inc., 1994, pp. 415–421.

[84] E. Reinhard and K. Devlin, "Dynamic range reduction inspired by photoreceptor physiology," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 11, no. 1, pp. 13–24, 2005.

[85] S. Pattanaik, J. Tumblin, H. Yee, and D. Greenberg, "Time-dependent visual adaptation for fast realistic image display," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 47–54.

[86] S. Pattanaik and H. Yee, "Adaptive gain control for high dynamic range image display," in *Proceedings of the 18th spring conference on Computer graphics*. ACM, 2002, pp. 83–87.

[87] M. Ashikhmin, "A tone mapping algorithm for high contrast images," in *Proceedings of the 13th Eurographics workshop on Rendering*. Eurographics Association, 2002, pp. 145–156.

[88] P. Choudhury and J. Tumblin, "The trilateral filter for high contrast images and meshes," in *ACM SIGGRAPH 2005 Courses*. ACM, 2005.

[89] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 257–266.

[90] Z. Rahman, D.J. Jobson, and G.A. Woodell, "Retinex processing for automatic image enhancement," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 100–110, 2004. [Online]. Available: http://link.aip.org/link/?JEI/13/100/1

[91] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 267–276, 2002.

[92] E. Reinhard, "Parameter estimation for photographic tone reproduction," *Journal of graphics tools*, vol. 7, no. 1, pp. 45–51, 2002.

[93] P. Ledda, L. Santos, and A. Chalmers, "A local model of eye adaptation for high dynamic range images," in *Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*. ACM, 2004, pp. 151–160.

[94] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range compression," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 249–256, 2002.

[95] G. Krawczyk, R. Mantiuk, K. Myszkowski, and H. Seidel, "Lightness perception inspired tone mapping," in *Proceedings of the 1st Symposium on Applied perception in graphics and visualization*. ACM, 2004, pp. 172–172.

[96] K. Smith, G. Krawczyk, K. Myszkowski, and H. Seidel, "Beyond tone mapping: Enhanced depiction of tone mapped hdr images," in *Computer Graphics Forum*, vol. 25, no. 3. Wiley Online Library, 2006, pp. 427–438.

[97] R. Mantiuk, K. Myszkowski, and H. Seidel, "A perceptual framework for contrast processing of high dynamic range images," *ACM Transactions on Applied Perception (TAP)*, vol. 3, no. 3, pp. 286–308, 2006.

[98] R. Mantiuk, S. Daly, and L. Kerofsky, "Display adaptive tone mapping," *ACM Trans. Graph.*, vol. 27, pp. 68:1–68:10, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360667

[99] (2011) pfstmo: Tonemapping library. Sourceforge. Open source Tonemapping library. See also pfstools and pfscalibration. [Online]. Available: http://pfstools.sourceforge.net/pfstmo.html

[100] S. Paris, S.W. Hasinoff, and J. Kautz, "Local laplacian filters: Edge-aware image processing with a laplacian pyramid," in *ACM Transactions on Graphics (SIGGRAPH)*, 2011.

[101] ——. (2011) Local Laplacian filters: Edge-aware image processing with a laplacian pyramid. Supplementary material and code for SIGGRAPH paper of same name. [Online]. Available: http://people.csail.mit.edu/sparis/publi/2011/siggraph/

[102] R.N. Clark. (2006, Sep.) The signal-to-noise of digital camera images and comparison to film. [Online]. Available: http://www.clarkvision.com/articles/digital.signal.to.noise/index.html

[103] (2011, Jan.) Technical application note TAN2008010: Overview of the ladybug image stitching process. Point Grey Research, Inc. Burnaby, Vancouver. [Online]. Available: http://www.ptgrey.com/support/downloads/documents/TAN2008010_Overview_Ladybug_Image_Stitching.pdf

[104] J. Blackman, "Omniveillance, google, privacy in public, and the right to your digital identity: a tort for recording and disseminating an individual's image over the internet," *Santa Clara L. Rev.*, vol. 49, p. 313, 2009.

[105] J. Segall, "Google street view: Walking the line of privacy-intrusion upon seclusion and publicity given to private facts in the digital age," *PGH. J. Tech. L. & Pol'y*, vol. 10, pp. 2–3, 2010.

[106] A. Frome, G. Cheung, A. Abdulkader, M. Zennaro, B. Wu, A. Bissacco, H. Adam, H. Neven, and L. Vincent, "Large-scale privacy protection in google street view," in *Computer Vision, 2009 IEEE 12th International Conference on*, 29 2009-oct. 2 2009, pp. 2373 –2380.

[107] A. Flores and S. Belongie, "Removing pedestrians from google street view images," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, 2010, pp. 53–58.

[108] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic imaging*, vol. 13, p. 146, 2004.

[109] Y. Hwang, J. Kim, and I. Kweon, "Determination of color space for accurate change detection," in *Image Processing, 2006 IEEE International Conference on*. IEEE, 2006, pp. 3021–3024.

[110] P. Blauensteiner, H. Wildenauer, A. Hanbury, and M. Kampel, "On colour spaces for change detection and shadow suppression," in *Proc. 11th Computer Vision Winter Workshop, Telc, Czech Republic*. Citeseer, 2006, pp. 87–92.

[111] R. Szeliski, "Image alignment and stitching: A tutorial," *Foundations and Trends in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2006.

[112] H. Zimmer, Jul. 2011, personal communication.

[113] (2011, May) Vertical bleeding or smearing from a saturated portion of an image. Point Grey Research, Inc. Knowledge Base Article 88. [Online]. Available: http://www.ptgrey.com/support/kb/index.asp?a=4&q=88

[114] S. Hasinoff, F. Durand, and W. Freeman, "Noise-optimal capture for high dynamic range photography," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 553–560.

[115] M. Granados, B. Ajdin, M. Wand, C. Theobalt, H.P. Seidel, and H.P.A. Lensch, "Optimal HDR reconstruction with linear digital cameras," in *CVPR*. IEEE, 2010, pp. 215–222.

[116] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition," *Progress in brain research*, vol. 155, pp. 23–36, 2006.

Index