Optical Splitting Trees for High-Precision Monocular Imaging

Morgan McGuire
Williams College
morgan@cs.williams.edu

Wojciech Matusik
MERL
matusik@merl.com

Billy Chen
Stanford
billyc@graphics.stanford.edu

John F. Hughes
Brown University
jfh@cs.brown.edu

Hanspeter Pfister
MERL
pfister@merl.com

Shree Nayar
Columbia
nayar@cs.columbia.edu

Abstract

The authors address the problem of designing and constructing efficient cameras that can simultaneously capture multiple pixel-aligned images for computational photography applications. They introduce a design paradigm and a prototype tool for computer-assisted design using that paradigm. Results are reported for actual cameras with up to eight imagers built using these techniques.

1. Introduction

Many computational photography applications require sets images that are captured simultaneously from the same viewpoint but have different image sensors and imaging parameters. In this paper, we address the problem of designing efficient multi-sensor cameras that can capture such images. Although for two- and three-sensor cameras ad-hoc designs are often effective, the design problem becomes challenging as the number of sensors increases. We demonstrate results on cameras created using our new design paradigm that contain as many as eight sensors.

Why are sets of pixel-aligned images of the same scene useful? Consider a set of images with different exposure times as an example. Short exposures will capture detail in the bright areas of a scene, while long exposures will reveal details in shadows that would otherwise be underexposed. Fitting an intensity curve to only the correctly-exposed images at each pixel fuses the set of images into a single high-dynamic range (HDR) image that captures detail everywhere. The output is 'computational' because the final image could not have been recorded by an isolated image sensor; it is produced by a battery of sensors and *computation.* Other popular applications that require include high-speed, super resolution, focus/defocus analysis, and multispectral video.

We say that the image sets contain multiple *monocular views* of the scene to distinguish them from stereo and other array cameras where the optical axis is different for each imager. Working with monocular views is ideal in many contexts because the images have direct pixel correspondance; except for calibration, they do not need to be warped and do not exhibit parallax artifacts with depth.

The cameras that can capture multiple monocular views differ from conventional cameras. A conventional camera contains a single objective (lens system) and imager. To

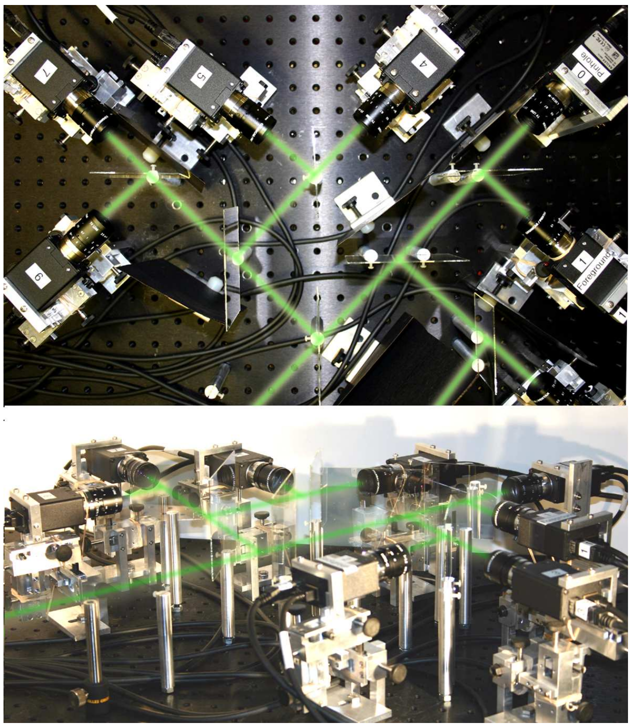


Figure 1. Top and side images of our generic eight-view splitting tree system, which can quickly be reconfigured to implement previous capture systems like like HDR and multispectral video, or to meet novel applications. The superimposed laser beam shows the recursively split optical path.

capture multiple views one makes an optical copy of the incident light using a beam splitter, like a half-silvered mirror. This allows two imagers to each observe the same view at half intensity. Inserting filters and other optical elements into the optical path between the beam splitter and the imager leads to different measurements at the corresponding pixels in each each imager. As described in the HDR example, these measurements can be combined into an image that represents more information (bits) than a single view, allowing for *high precision* measurements. In this paper, we consider optical systems that form *optical splitting trees*, where the the optical path topology takes the shape of a tree due to recursive beam-splitting.

Designing optical splitting trees is challenging when many views with specific spectral properties are required. We introduce a manual design paradigm for optical splitting trees and a computer-assisted design tool implements that design paradigm to create efficient splitting-tree cameras. The tool accepts as input a specification for each view and a set of weights describing the user's relative affinity for efficiency, accuracy, and cost. The weights determine the objective function of an optimizer. The output is a splitting tree design that implements the input specification, and an analysis of the efficiency of each root-to-leaf path. Automatically designed trees appear comparable to those designed by hand; we even show some cases where they are superior.

Capture of multiple monocular views has previously been demonstrated in various contexts that we review, but rarely have more than three simultaneous views been captured. We have built the configurable optical splitting tree system shown in Figure 1, which captures up to eight views, and implemented several popular applications: HDR, multifocus, high-speed, and hybrid high-speed multispectral video, assisted by our optimizer.

2. Related Work

2.1. Beam Splitters

Both prisms and half-mirrors are popular beam splitting mechanisms. They can be constructed to split light into two or more paths, and the ratio of intensities directed to each path at each wavelength can be adjusted. The most common and economical element is a half-silvered plate mirror. A drawback of plate mirrors is that their orientation must be calibrated relative to the optical path. In contrast, sensors placed immediately against the sides of a splitting prism are automatically registered up to a 2D translation. In the case of 3-CCD cameras, a dichroic prism that separates light by wavelength is often used to capture three copies of the image, each with a different spectral band. Prisms have also been used for HDR imaging [6].

Our implementation places beam splitters between the lens and the scene, which enables us to use separate lens parameters for each sensor. An alternative is to split the light in between the lens and the sensor [11]. That alternative shares a single lens over all sensors, which simplifies lens calibration and reduces lens cost, but makes calibration and filter changes more difficult.

McGuire et al. [7] use a beam splitter camera with three views for matting. They report that it can be extended to exhaustively sample various sampling parameters. We extend these ideas with a detailed discussion and implementation of a full, calibrated eight-view system, analysis methods, and *sparse* parameter sampling that allows us to create views with completely independent characteristics. We demonstrate more efficient and economical camera designs for their application in Section 6.3.

2.2. Pyramid Mirrors

Another interesting way to create multiple copies uses a pyramid mirror placed behind the lens [1, 5]. (Placing a mirror pyramid *in front* of the lens, *e.g.* as in [13], creates panoramic field of view that is unrelated to our work.) This arrangement creates a very compact optical path, but has some drawbacks. It requires a large aperture, which leads to a narrow depth of field and limits the situations to which it may be applied. It is also non-trivial to divide light intensity unevenly between the image copies, as might be desirable for HDR. Furthermore, the edges between the individual mirrors cause radiometric falloffs as discussed in [1]. Even after calibration these fall-offs reduce the effective dynamic range of each view. The defocus point spread function from such a camera is a differently oriented triangle in each view, instead of a disk as in a beam splitter camera. This makes it difficult to fuse or compare images in which objects are at different depths; objects outside the depth of field appear not only defocused but also shifted away from their true positions.

2.3. Alternatives

For flat scenes and scenes far ($>$ 10m) from the camera, there are neither parallax nor view-dependent effects. In those cases the calibration problem is comparatively easy since the optical centers of the sensors need not be aligned. Dense arrays of side-by-side sensors, *e.g.*, [14], have captured multiple approximately-monocular views for such cases. Arrays capture much more light than a beam splitter system. However, a beam splitter system can capture nearby and deep scenes, and offers the possibility of sharing optical elements like filters over multiple sensors.

One can use a mosaic of filtered CCD pixels to sample multiple parameters in a single image. The Bayer mosaic tiles per-pixel band-pass filters, sampling three wavelengths with a single monochrome sensor. Recently, filter mosaics have been proposed for sampling other parameters

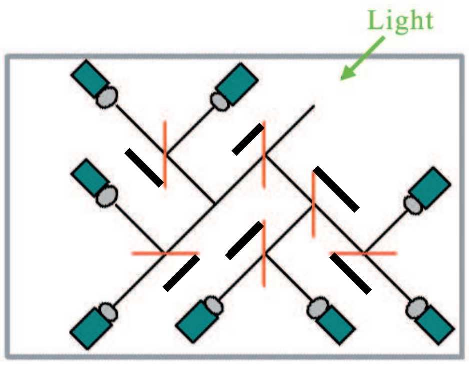


Figure 2. Physical layout of a balanced eight-view splitting tree.

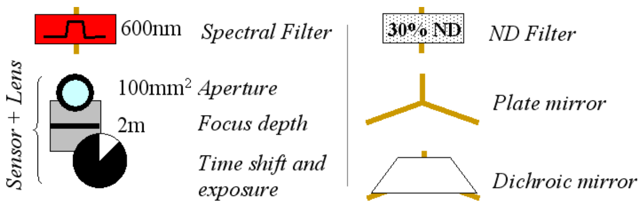


Figure 3. Lexicon of symbols used in our abstract tree diagrams.

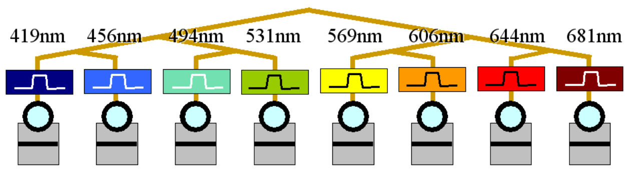
with high precision (see [9, 10]). This approach can be implemented compactly and requires no calibration (once manufactured), making it ideal for many applications. The drawback is that it trades spatial resolution for resolution along other imaging dimensions. Such a system also makes it difficult to experiment with aperture and timing effects, which are explored in this paper.

Previous commercial optical design systems like SYNOPSYS, ADOS, and ZEMAX emphasize the tracing of rays through lenses. To the best of our knowledge, none significantly address the issues of sampling the plenoptic function through splitting and spectral response that we discuss.

3. Optical Splitting Trees

Optical splitting trees are schematic representations of a tree-topology filter systems. The edges are light paths and the nodes are optical elements. Nodes with a single child represent filters and lenses. Nodes with multiple children are beam splitters. Leaf nodes are sensors. The plenoptic field enters the system at the root. The physical path length to each sensor's optical center is identical. However, the *tree-depth* of a sensor (the number of internal nodes between it and the root) may differ.

Figure 2 shows a schematic of a full binary splitting tree, viewed from above, where light enters on the upper right. This layout packs components into a small form factor with no optical path occluded. The thick black lines are light baffles preventing the reflective child of each splitter from imaging stray light. Further abstracting the structure and using the symbols in Figure 3, we can emphasize the most significant elements. We know that all paths from the root



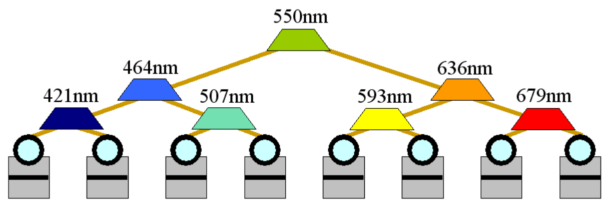


Figure 4. (top) Economical band pass and (bottom) efficient dichroic mirror multispectral trees.

to the leaves have the same physical length, so the representation need not preserve distances. Angles are an artifact of building the physical system and also need not be represented. We are left with an abstraction where only graph topology is of significance. The tree has a branch factor of at most two when the beam splitters are half-mirrors (as in our implementation). Other splitting elements can produce higher degree nodes. Subtrees of beam splitters with no intervening filters can be collapsed to a single node representation with many children, as in Figure 13. This representation also abstracts the nature of the of splitting element that is being employed.

Figure 4(bottom) shows a balanced multispectral splitting tree. Each beam splitter is a dichroic mirror, which divides half the remaining visible spectrum among its children so that no light is lost, but manufacturing large dichroic mirrors is expensive. Figure 4(top) schematically shows a less efficient design using readily available band-pass filters. Each path receives about 1/8 of the incident light, which is band-pass filtered immediately before the lens. Note that both designs are useful; the splitting tree concept makes it easy to switch between them if necessary.

Many applications require a balanced binary tree, in which each sensor has the same tree depth and the beam splitters divide incident light evenly between their children. In others (like HDR) it is useful to unbalance the tree. We may do so either by using beam splitters with uneven division ratios, or by creating a structurally unbalanced tree where the sensors' tree-depths vary.

4. Assisted Design

For a complex camera, manual design becomes challenging. This is in part because elements like beam splitters and filters have non-ideal responses (see Figure 5). Manually accounting for this spectral distortion is difficult, so people are likely to design sub-optimal cameras, especially in the experimental stage. Another complicating factor is that multiple trees may satisfy a specification but differ in other ways, *e.g.* Figures 7 and 4 demonstrate cost and efficiency tradeoffs.

We now describe our method for an assisted design tool that creates a splitting trees to meet a formal specification. It addresses the complexity of designing large trees and can balance non-ideal components against each other. It also takes into account efficiency, cost, and accuracy as defined later. The design process contains two phases. The first deterministically constructs an inefficient, expensive tree that is close to the specification and then applies a deterministic series of simplifications to reduce the cost. The second phase is an optimizer that performs short random walks through the space of all possible trees, searching for similar but better trees. It continues until explicitly terminated by the user. Our implementation is in Matlab, using trees from the Data Structures toolbox. All results were computed in under an hour, although usually the first minute brought the objective function within 10% of the peak value and the remaining time was spend addressing minor spectral deviations.

4.1. Input Specification

Our algorithm takes a set of design specifications $\{\hat{X}, \hat{I}, \hat{Y}\}$ and objective function weights $(\alpha, \beta, \delta, \epsilon, \gamma)$ as input. It also contains a database of components from a real optics catalog[1]The design specification is described below; the weights are discussed in section 4.3 during presentation of the objective function.

For each view v, the user specifies:

$\hat{X}_v[\lambda]$	The desired spectral response curve, as a unitless vector indexed by wavelengths.
$\hat{I}_v$	The (unitless, scalar) importance of this sensor relative to others.
$\hat{Y}_v[p]$	A vector of the scalar imaging parameters (p): aperture area, exposure time, polarization sensitivity, time shift, horizontal and vertical sub-pixel shift, focal length, and focus depth.

In this notation, a hat over the variable distinguishes a specification variable from its counterpart that is adjusted during optimization. The subscript v denotes the sensor that a variable describes.

Note that no layout hints are provided with the specification; the tool begins from scratch.

4.2. Efficiency(q)

Efficiency is a desirable quality in a camera; this section rigorously defines the efficiency q of a camera. Efficiency is

[1]Edmund Optics Catalog 2005

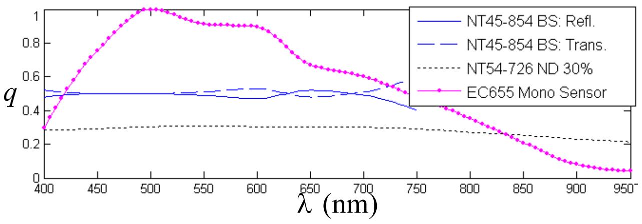


Figure 5. Actual efficiency curves for nominally 'uniform' components: beam-splitter #854, neutral-density filter #726, and a Prosilica CCD. Because these curves are not ideal flat, horizontal lines, naive manual design loses efficiency. In contrast, our design tool considers the actual curves and can exploit imperfections in the filters to boost overall light efficiency.

not an explicit part of the specification but is instead a term in the optimization process.

The specification $\{\hat{X}, \hat{I}, \hat{Y}\}$ dictates how the tree distributes light, but the scale is necessarily relative because at input time the light loss inherent in the design is unknown. Likewise, each component in the catalog must be annotated with a curve describing its transmission (for a filter), split ratio (for a beam splitter), or digital output sensitivity (for a sensor) at several wavelengths. Figure 5 shows examples of these curves.

We call this curve the *quantum efficiency* of a component. We represent it with a vector q describing the ratio of output to input light at a node, sampled at twelve wavelengths between 400nm and 950nm. Because our components are passive, $q[\lambda] \leq 1$. For a view v, let $q_v = \prod q_i$ over every component i on the path from v to the root. Let scalar $\bar{q}_v$ be the mean efficiency of that view with respect to λ, and let $\bar{q}_{tree} = \sum_{views} \bar{q}_v$, denote the efficiency of the entire camera. With these definitions, $\hat{X}_v = q_v / \bar{q}_v$ specifies the shape of the desired response and $\hat{I}_v = \bar{q}_v / \bar{q}_{tree}$ is the fraction of measured light captured by v.

4.3. Objective Function

The optimizer seeks camera designs that maximize the goals of efficiency, accuracy, and cost. The objective function is the weighted sum of expressions representing each of these goals,

$$\begin{aligned} obj(tree) = \quad & \alpha \bar{q}_{tree} && (1)\\ -\textstyle\sum_{views} (\quad & \left|\beta\left(X_v - \hat{X}_v\right)\right|^2 && (2)\\ & +|\delta\left(I_v - \hat{I}_v\right)|^2 && (3)\\ & +\left|\vec{\epsilon}\left(Y_v - \hat{Y}_v\right)\right|^2) && (4)\\ -\gamma \textstyle\sum_{nodes} c_i^2 \quad & . && (5) \end{aligned}$$

These expressions drive the optimizer as follows:

Efficiency (1): maximize the total light measured;

Spectral accuracy (2): minimize the difference from the color specification;

Relative importance (3): respect the specified splitting ratios;

Parameter accuracy (4): minimize difference from the specified pixel shift, temporal shift, polarization, motion blur, and defocus; and

Economy: (5): minimize the dollar cost.

Each expression is quadratic to create stable maxima. Recall that X and Y are vectors, so $|\cdot|^2$ is a dot product, and that the hats denote specification variables.

Greek-letter variables are user-tunable weights, which may vary from zero (which ignores a term) to arbitrarily large positive numbers (which demand that the optimizer exactly meet the specification). They are all scalars except for $\vec{\epsilon}$, which is a vector so that each imaging parameter may be weighted independently.

We choose initial weights so that each of the five expressions has approximately the same magnitude. (Note that this depends on the units selected for Y). Result quality is most sensitive to α because preserving accuracy introduces filters that absorb light. Other weights can vary within a factor of two without affecting the output, since cost and accuracy are less contentious when many filter choices are available.

4.4. Deterministic Phase

The goal of this phase is to construct a tree that accurately meets the view specifications. To simplify the process, economy and efficiency are left unconstrained. The system first groups the views into separate binary trees of similar $\hat{X}$ to increase the later likelihood of shared filters and then links those trees into a single large tree.

All splitters are "50R/50T" half-mirrors at this stage. To satisfy the $\hat{X}$ values, the system evaluates the actual X at each leaf and introduces band pass filters immediately before the sensors to optimize the spectral accuracy term. It then sets all parameters as dictated by $\hat{Y}$. Finally, it evaluates I at each leaf and inserts neutral-density filters until the importance accuracy term is optimized.

4.5. Search Phase

From the deterministically computed tree we search for steps in the design space that increase obj using the uphill simplex method. Each step begins with a randomly chosen transformation. Because many transformations require parameter changes to be beneficial, the Y vectors are adjusted to increase spectral and importance accuracy before obj is evaluated for the altered tree.

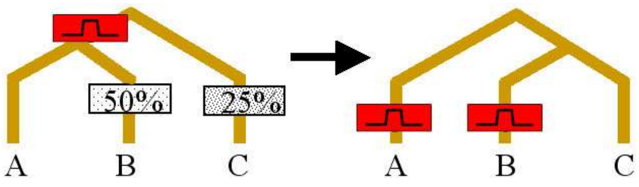


Figure 6. This 'right rotation' transformation at the root reparents node B. Both designs create the same measurement, but the design on the right is twice as efficient. However, this is not always advantages. After the transformation, the spectral filter on A and B is duplicated, which might make the system too expensive.

Several transformations preserve x and I while potentially reducing the cost of the tree or increasing $\bar{q}_{tree}$. These "tree identities" are:

1. No transformation (allows Y change without tree change).
2. If the same filter appears on both children of a beam splitter, move it to the parent of the splitter.
3. Replace a chain of filters with a single, equivalent filter.
4. Reorder the filters in a chain (tends to encourage #2 and #3).
5. Node Rotation: *e.g.*, in Figure 6.
6. Replace a splitter and filters on its children with a splitter whose R/T ratio that approximates the filter ratio.

Other transformations can change X and I and are therefore less likely to improve the tree. Nonetheless, they provide reachability to the entire design space and therefore allow the optimizer to theoretically find the best tree given infinite time. These include:

1. Add, remove, or change a filter at random.
2. Rotate a sub-tree right or left without adding filters.
3. Replace a beam-splitter at random.
4. Swap two sub-trees.

5. Experimental System

To test our design framework we built a physical configurable splitting tree system with eight computer vision sensors that uses 100×100mm half-mirror and hot-mirror beam splitters (see Figure 1; the black baffles in the top view were removed for bottom view).

The sensors are 640×480 Bayer-filter A601fc color cameras and monochrome A601f cameras by Basler. Each sensor is equipped with a Pentax 50mm objective. The tree exactly fits on a $2 \times 2\text{ft}^2$ optical breadboard with $1/2''$ in hole spacing. The sensors are connected to a single 3 GHz P4 PC using the FireWire interface. We wired the hardware shutter trigger pins to each of the eight data pins of the PC parallel port, which can be precisely controlled by writing bit masks to that port through a special Win32 driver[2].

[2]http://www.logix4u.net/inpout32.htm

5.1. Calibration

The difficulty of calibration increases with the number of elements. Sensors that share an optical center are also more difficult to calibrate than array systems where the views are not expected to align perfectly.

We first orient the half-mirror beam splitters at 45° to the optical axis. To orient these, we place a lens cap over each camera and shine a laser along the optical axis to illuminate a single point near the center of each lens cap. Working through the splitting tree from the root to the leaves, we rotate the beam splitters until each dot appears exactly in the center of the lens cap.

Second, we construct a scene containing a nearby target pattern of five bulls eyes on transparent plastic and a distant, enlarged pattern on opaque poster board so that the two targets exactly overlap in view 1. We then translate all other sensors until the target patterns also overlap in their views. This ensures that the optical centers are aligned.

Third, we compute a software homography matrix to correct any remaining registration error. We find corresponding points in 3D by filming the free movement of a small LED light throughout the scene. Let C_s be the $N \times 3$ matrix whose rows are homogeneous 2D positions, *i.e.* $[x\ y\ 1]$, of the light centroid at subsequent frames in view number s. The transformation mapping pixels in view 1 to those in view s is

$$H_s = \text{argmin}(|H_s C_s - C_1|^2) = C_1 C_s^{\dagger}, \qquad (6)$$

where $\dagger$ denotes a pseudo-inverse. We solve this system by singular value decomposition because it is frequently ill-conditioned. For color calibration, we solve the corresponding system in color space using pixel values sampled from a Gretag Macbeth color chart instead of 2D positions.

6. Applications and Results

We implemented various video capture applications using a mixture of data acquisition trees that were hand-designed using the framework and ones produced automatically by the optimizer. Working from scratch, it often takes several days to assemble and calibrate a new computational photography camera for a single application. Because our hardware system is configurable and most splitting trees form subsets of the full tree that we pre-build on the optical table, we can configure for different applications comparatively quickly. For each of the examples described here we were able to reconfigure and calibrate the system in about two hours, even when outside the laboratory. Paired with the assisted design tool, this allows for much greater experimental flexibility than we had previously enjoyed.

In each example, the result figures and videos demonstrate accurate color, temporal, and spatial image capture and registration across many more monocular views than in previous work.

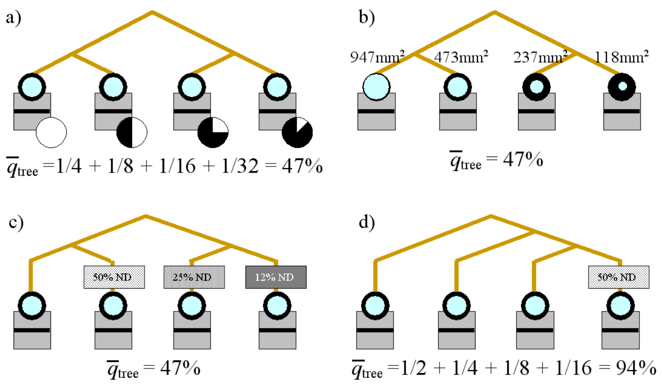

a)
$\overline{q}_{\text{tree}}$ =1/4 + 1/8 + 1/16 + 1/32 = 47%
b)
947mm²
473mm²
237mm²
118mm²
$\overline{q}_{\text{tree}}$ = 47%
c)
50% ND
25% ND
12% ND
$\overline{q}_{\text{tree}}$ = 47%
d)
50% ND
$\overline{q}_{\text{tree}}$ =1/2 + 1/4 + 1/8 + 1/16 = 94%

Figure 7. HDR video trees that vary a) exposure, b) aperture, c) filters, and d) structure. The latter is the most efficient.

The deterministic phase always produces a viable design, so optimization will technically never fail. However, in about two out of ten cases, it takes longer to adjust the weights than it would to simply adjust the output of the deterministic phase manually. In its current form, the optimizer is therefore most useful when applied to designs with many sensors, significantly non-uniform components, or tricky spectral constraints.

6.1. High Dynamic Range

Figure 7 shows several splitting trees for a simple high dynamic range camera where the relative intensities observed by the views are powers of two. We designed these by hand based on previous work and verified that with appropriate weights the optimizer rediscovered equivalent structures. In (a), a large economy weight γ gives the inexpensive variable exposure solution [4]. The drawbacks of that approach are inconsistent motion blur between cameras and low $\bar{q}_{tree} = 15/32$ efficiency. Increasing the exposure weight and decreasing the aperture weight in $\vec{\epsilon}$ leads to (b), where the aperture varies. Now motion blur is correct but the depth of field varies and $\bar{q}_{tree}$ is still low.

An alternative is to use neutral-density filters as in (b), similar to Mitsunaga et al. [8]. Compared to Debevec et al.'s method, this corrects both motion blur and depth of field but not efficiency. Our optimizer did not re-discovered this approach–instead it found a **better** design!

Tree (d) has $\bar{q}_{tree} = 30/32$. Instead of blocking light with the iris, shutter, or filters, it redirects excess light at a sensor to other sensors that can measure it. Aggarwal and Ahuja [1] mention this design, but it produces asymmetric point spread functions on their pyramid camera and was never implemented.

Figure 8 shows results from two HDR experiments. In sequence number one (top), the actor is brightly lit and the city lights are dim in the background. In sequence number

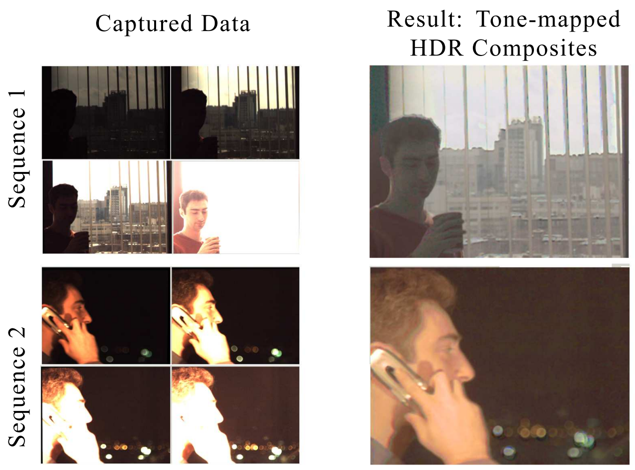


Figure 8. Frames from two HDR sequences. Images on the left are four simultaneously captured views. The large images on the right are the corresponding tone-mapped composites.

two (bottom), the actor is inside a dark office and the sky and city in the distance are bright. On the right are resulting tone-mapped HDR images. These combine frames from four different sensors to keep all scene elements visible and within the dynamic range of the display.

6.2. Multiple Focus and Defocus

Images focused at multiple depths can be used to recover depth information [11, 2, 15] and to form images with an infinite [12] depth of field. Many systems (*e.g.*, [11]) split the view behind the lens. Splitting in front of the lens allows us to vary not only the location but also the depth of the field by changing the aperture.

To capture images with varying depths of field like those in Figure 9 we use a full binary tree with with eight cameras. Each sensor is focused at a different depth, ranging from 20cm from the optical center (about 4cm from the first beam splitter) to 20m (effectively infinity). We use wide $f/1.4$ apertures for a narrow depth of field on each sensor.

Figure 9 shows an artificially wide depth of field achieved by a weighted sum of each view. The weight at each pixel is proportional to the local contrast (luminance variance) in the view, squared.

6.3. Matting and Other Designs

Two matting algorithms from the computer graphics literature use custom cameras that can be expressed within our framework. We compare the originally published designs against new ones created by our optimizer from their specifications.

McGuire et al. [7] capture three monocular views, each with focal length $f = 50$mm and each at equal importance: a pinhole camera with a $f/12$ aperture, and two $f/1.6$ views focussed at different depths. Their design uses

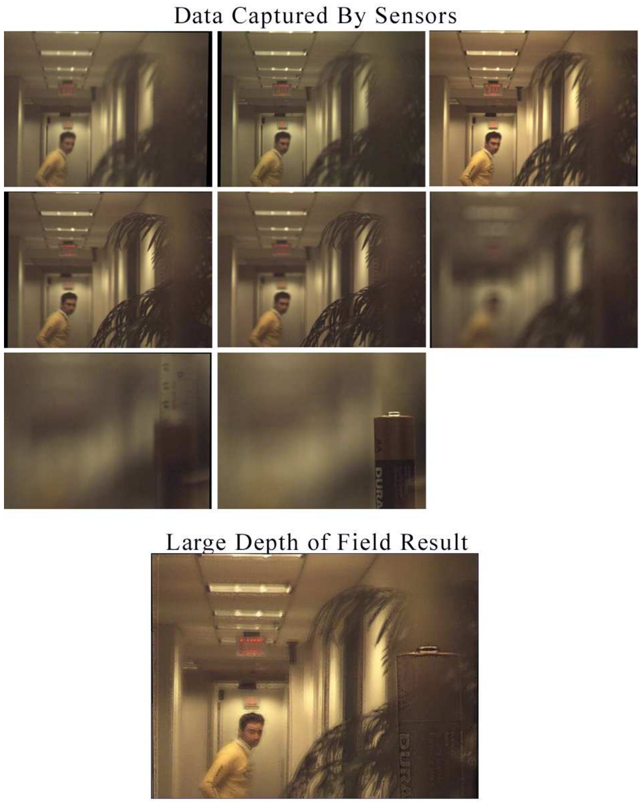


Figure 9. Multifocus images. Eight simultaneous views with different focus depths, and a fused infinite depth-of-field result.

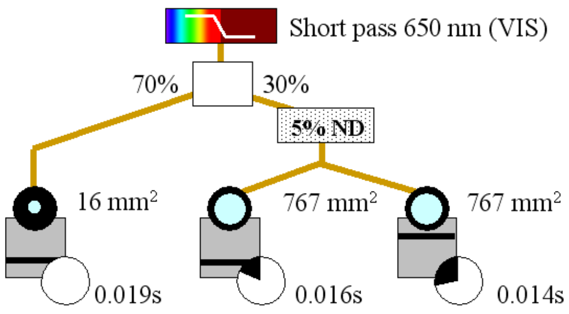
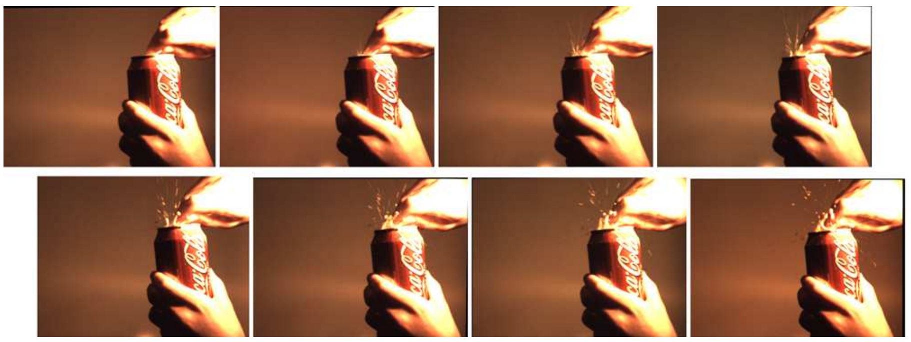


Figure 10. Optimized matting camera based on McGuire et al. [7].

two "50R/50T" beam splitters and two filters.

Our optimizer created the alternative tree in Figure 10. Compared to the original, this design achieves higher efficiency through a "70R/30T" beam splitter; a single, weaker neutral-density filter shared across two views to reduce cost; and a short pass filter to attenuate the color sensors' undesirable IR response. The optimizer adjusted exposures slightly to compensate for imperfections at the "50R/50T" beam splitter, where it chose a low quality component to reduce cost. During optimization the short pass filter separately originated at each leaf and was propagated to the root.

We performed a similar experiment on Debevec et



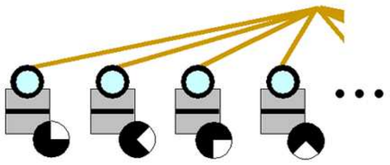


Figure 11. 240 fps video of a soda can opening. Each of the eight sequential frames shown was captured by a different sensor. The tree is on the right; note the 1/120s overlapping exposures, longer than is possible for a single-sensor high speed camera.

al. [3]'s matting system that uses a half-mirror and an IR filter to capture two monocular views. The optimizer simply replaced the half mirror with a hot mirror to increase efficiency at no additional cost.

The matting cameras are simple. Figure 12 shows the tree computed for an arbitrary complex specification. The optimizer correctly created an efficient HDR (c,d,e) subtree. However, it failed to place a hot mirror between (a) and (b), probably because we weighed accuracy much higher than efficiency for this test. The six plots show how well it did match the desired accuracy; it even chose between different *brands* of neutral-density filters based on their q-curves.

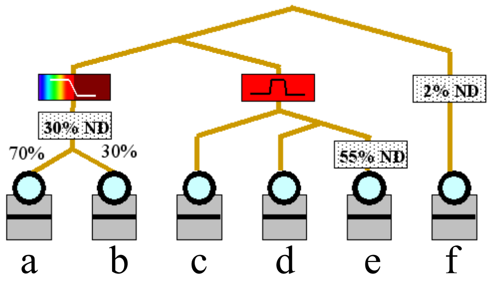
6.4. High Speed

Figure 11 shows eight frames from a high speed sequence of a soda can opening and the capture tree used. Each frame has an exposure time of $1/120$s and the entire sequence is captured at 240fps, so the views overlap temporally. This gives $\bar{q}_{tree} = 1/4$, which is lower than the $\bar{q}_{tree} = 1$ for an array like the one by Wilburn et al. [14]. The advantage of our approach is that the sensors share an optical center for accurate capture of scenes with depth variation and view-dependent effects.

6.5. Multimodal High Speed

HDR, high speed, etc. are sampling strategies. They are useful for building high-level applications, like surveillance in an HDR environment. It is natural to build hybrid sampling strategies, which are easy to express and experiment with within our splitting tree framework.

We use the configuration from Figure 13 designed by the optimizer to capture hybrid high-speed visible/IR video. A hot-mirror directs IR down the right sub-tree and visible light down the left sub-tree. Each subtree has four cameras with temporal phase offsets, so the entire system yields 120fps video with four spectral samples. Figure 14 shows a frames from a sequence in which a person catches a tumbling IR remote control and then transmits at the camera.Because the configuration captures four spectral samples at 120fps, the high-frequency IR pattern transmitted by the remote is accurately recorded, as is the fast tumbling motion at the beginning of the sequence.



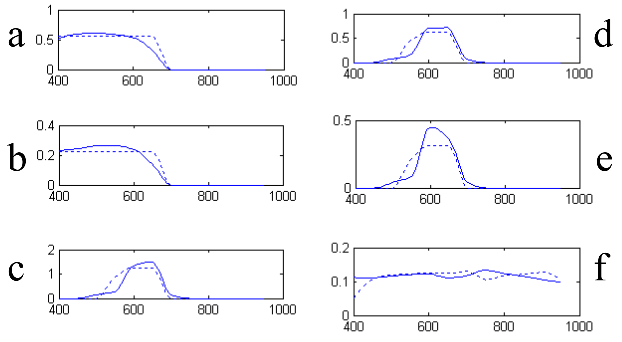


Figure 12. Plots of $(I \cdot x)$ vs. λ nanometers for a complex camera. Dashed lines are the specification, solid lines are the final design. The design tool independently matched each spectral efficiency curve close to the specification, taking into account imperfections in the actual filters.

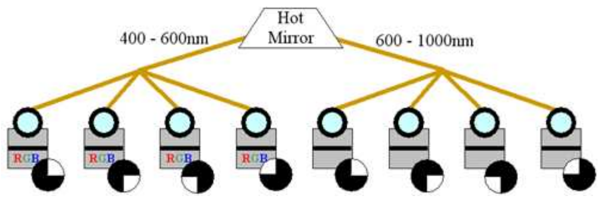


Figure 13. Hybrid high speed, multimodal visible + IR camera.

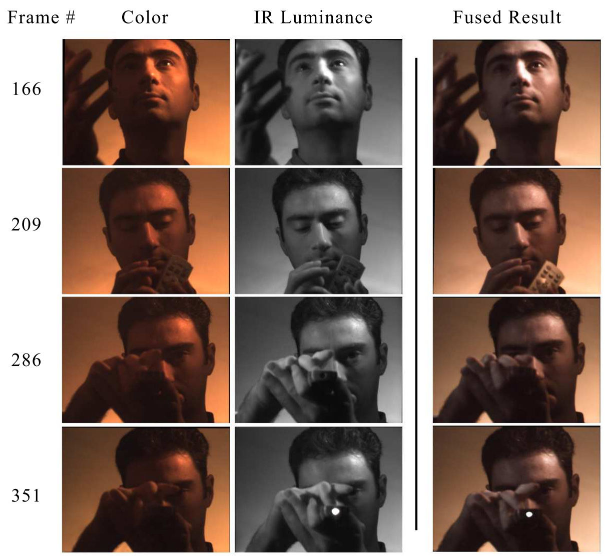


Figure 14. Frames of high-speed video with visible and IR channels. Note the IR-only illumination on the subject's right and at the LED on the remote control.

7. Conclusions and Future Work

We presented a framework useful for manual camera design and an algorithm for automatic design. We also implemented a configurable system that samples multiple parameters per frame and demonstrated its utility. Using our framework and this system, high-precision imaging applications become easier to develop and produce results of comparable or better quality than alternative solutions.

Manual designs are easily understood but rely excessively on the notion of ideal components. Automatically designed trees contain surprising and complex element choices. These micro-balance the true response curves of components and are frequently more efficient.

As future work,we are investigating multispectral HDR for surveillance, alternative multi-focus approaches for matting, and multispectral high-speed for material testing in the context of splitting trees. Another area of future work is addressing the limitations of the current automatic design approach to find more efficient and general designs. For example, element cost should be a function of tree depth, since filters closer to the root must be larger to fill the field of view. Genetic algorithms are naturally suited to tree combination and should be more effective than our uphill simplex.

The optical elements form a filter system that terminates at digital sensor. In an application, this is always followed by a digital filter system. The next step is to simultaneously design both the optical and software filter systems.

References

[1] M. Aggarwal and N. Ahuja. Split aperture imaging for high dynamic range. *International Journal of Computer Vision*, 58(1):7–17, June 2004. 2, 6

[2] S. Chaudhuri and A. Rajagopalan. *Depth from Defocus: A Real Aperture Imaging Approach*. Springer-Verlag, 1998. 7

[3] P. Debevec, A. Wenger, C. Tchou, A. Gardner, J. Waese, and T. Hawkins. A lighting reproduction approach to live-action compositing. *ACM Trans. on Graphics*, 21(3):547–556, July 2002. 8

[4] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH*, pages 369–378. ACM Press, 1997. 6

[5] R. P. Harvey. Optical beam splitter and electronic high speed camera incorporating such a beam splitter. United States Patent US5734507, 1998. 2

[6] E. Ikeda. Image data processing apparatus for processing combined image signals in order to extend dynamic range. U.S. Patent 5801773, Sept. 1998. 2

[7] M. McGuire, J. Hughes, W. Matusik, H. Pfister, and F. Durand. Defocus matting. *Trans. on Graphics*, 24:567–576, 2005. 2, 7

[8] T. Mitsunaga and S. Nayar. High dynamic range imaging: Spatially varying pixel exposures. In *CVPR*, volume 1, pages 472–479, 2000. 6

[9] S. Nayar and T. Mitsunaga. High dynamic range imaging: Spatially varying pixel exposures. In *CVPR*, pages I: 472–479, 2000. 3

[10] S. Nayar and S. Narasimhan. Assorted pixels: Multi-sampled imaging with structural models. In *ECCV*, page IV: 636 ff., 2002. 3

[11] S. K. Nayar, M. Watanabe, and M. Noguchi. Real-time focus range sensor. *Trans. on PAMI*, 18(12):1186–1198, 1996. 2, 7

[12] M. Subbarao. Parallel depth recovery by changing camera parameters. In *ICCV*, pages 149–155, December 1988. 7

[13] K.-H. Tan, H. Hua, and N. Ahuja. Multiview mirror pyramid cameras. *Trans. on PAMI*, 26(7):941–946, July 2004. 2

[14] B. Wilburn, N. Joshi, V. Vaish, M. Levoy, and M. Horowitz. High speed video using a dense camera array. In *CVPR*, June 2004. 2, 8

[15] Y. Xiong and S. Shafer. Depth from focusing and defocusing. In *DARPA93*, pages 967–, 1993. 7